[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 15/16

# [12]发明专利申请公开说明书

[21]申请号 99808006.3

[43]公开日 2001年10月24日

[11]公开号 CN 1319209A

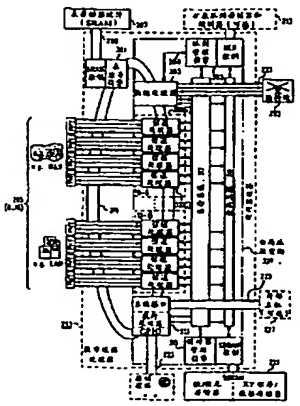
[22]申请日 1999.5.7 [21]申请号 99808006.3
[30]优先权
[32]1998.5.8 [33]US [31]60/084,706
[32]1998.10.27 [33]US [31]60/105,823
[86]国际申请 PCT/US99/10002 1999.5.7
[87]国际公布 WO99/59078 英 1999.11.18
[85]进入国家阶段日期 2000.12.28
[71]申请人 摩托罗拉公司
地址 美国伊利诺斯
[72]发明人 托马斯·B·布莱特曼
安德鲁·T·布朗 约翰·F·布朗
詹姆斯·A·法雷尔 安德鲁·D·芬克
戴维·J·赫塞克 爱德华·J·迈克莱伦
唐纳德·A·普莱尔 马克·A·桑基
保罗·施米特
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 付建军

权利要求书5页 说明书101页 附图页数47页

[54]发明名称 数字通信处理器

[57]摘要

一个集成电路(203),用于处理一般的数据流,并且用于处理特 殊的包流。这个集成电路包括一些包处理器(307,313,303),一个 表搜寻引擎(301),一个队列管理引擎(305)和一个缓冲器管理引擎 (315)。这些包处理器包括一个接收处理器(421),一个发送处理器(427)和一个 RISC 核心处理器(401),所有这些处理器均是可以被 编程的。这个接收处理器和这个核心处理器进行合作来接收和路由正 在被接收的包,并且这个核心处理器和这个发送处理器进行合作来发 送包。通过使用来自表搜寻引擎的信息决定这个队列管理引擎中、将 接收描述这个被接收包的负荷的一个描述符217 的一个队列215,来 完成这个路由。这个发送处理器从一个队列中读取描述符,并且发送 在这个描述符中所描述的负荷。这个核心处理器,发送处理器,和接 收处理器并行进行工作。这些包处理器中的、队列管理引擎中的和缓 冲器管理引擎中的局部存储器和寄存器是一个全局地址空间(321)的 一部分。这些包处理器包括用于处理在串行媒质中所接收的包的串行 处理器(307)和至少一个并行处理器(303)。这个并行处理器可以被 用于将这个集成电路连接到另一个相同类型的集成电路,连接可以是 使用一个总线,或者使用一个交换构造。通过设置全局存储器中寄存 器中的比特,这些包处理器可以被进行进一步配置。配置包括串行包 处理器的组合,在一个串行包处理器中重新循环一个数据流,旁路这 个接收或者发送处理器的部件,并且配置一个串行包处理器的 I/O 管 脚来处理不同的传输媒质。



ISSN1008-4274

# 权 利 要 求 书

1. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，并且包括：

一个包括指令的可被写入的指令存储器；和

一个接收处理器，连续地执行特定的指令来处理从这个数据流输入所接收的数据流和/或者

一个发送处理器，连续地执行特定的指令来处理需要被输出到数据流输出的数据流。

2. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，并且包括：

一个对这个数据流处理器是局部的、可被写入的局部存储器，

多个局部存储器属于可以被任何一个数据流处理器所访问的一个全局地址空间。

3. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连；和

一个上下文处理器，对从正在处理一个数据流的一给定数据流处理器所接收 的信息作出响应以产生关于这给定数据流的上下文的信息，并且将这个上下文信息提供给这个处理器；

这个给定数据流处理器使用这个上下文信息来处理这个数据流。

4. 如权利要求 3 所提出的这个集成电路，其中：

这个上下文处理器接收这个信息，并且通过一个总线来提供上下文信息，在这个总线上，对数据流处理器中的每一个和上下文处理器

的延迟有一个上限。

5. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流，一个数据流包括控制数据和负荷；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，一个被接收的数据流正在被进行处理以提取这个控制数据和负荷，并且一个被发送数据流正在被处理以将控制数据添加到这个负荷上；

一个缓冲器管理器，提供用于保存负荷的缓冲器的地址，并且对带一个缓冲器地址的一个写入操作作出响应，将负荷写入到这个被寻址的缓冲器，并且对带一个缓冲器地址的一个读取操作作出响应，以从这个被寻址的缓冲器读取负荷；和

一个队列管理器管理负荷描述符的队列，并且每一个描述符包括至少一个缓冲器地址，这个队列管理器通过将一个排队命令所提供的一个描述符在这个命令中所规定的一个队列中进行排队来对这个排队命令作出响应，通过在一个出列命令所规定的这个队列中取出一个描述符来对这个出列命令作出响应，

一个数据流处理器通过使用所接收的一个数据流的负荷和这个缓冲器管理器所提供的一个地址来对缓冲器管理器执行一个写入操作，通过使用包括这个地址的一个描述符来执行一个排队操作并且通过执行从一个队列中进行取出的操作来发送一个数据流，来对这个所接收的数据流作出响应，并且使用作为在对缓冲器管理器的一个读取操作中一个从队列中进行取出操作的一个结果而获得的地址，使用从这个缓冲器管理器所接收的负荷来产生一个数据流，并且发送所产生的数据流。

6. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连；和

一个组合特定数据流处理器的组合器，以使被组合的数据流处理器可以在处理一个数据流时进行合作，这个组合器包括

在被组合的数据流处理器之间的、可配置的互联；

一个可配置的操作协调器，用于协调被组合数据流处理器之间的操作；和

一个配置器，用于按照组合这些数据流处理器的需要来规定可配置的互联和可配置的操作协调器。

7. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，并且包括

一个控制数据处理器，

一个接收处理器，处理从这个数据流输入所接收的数据流，和/或者

一个发送处理器，用于处理需要被输出到这个数据流输出的数据流，和

数据结构，被控制数据处理器，接收处理器和/或者发送处理器所共享，并且包括被接收处理器和/或者发送处理器和控制数据处理器对接收处理器和/或者发送处理器和控制数据处理器采用流水线的方法处理一个数据流进行协调所使用的信息。

8. 一个集成电路，包括：

多个数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，并且包括

一个接收处理器，用于串行地处理从这个数据流输入所接收的数据流，和/或者

一个发送处理器，用于串行地处理需要被输出到数据流输出的数据流，

这个接收处理器和/或者发送处理器具有多个处理部件，并且可以

被配置成在处理数据流时旁路一个或者多个部件。

9. 一个集成电路，包括：

多个串行数据流输入和/或者输出，它们接收和/或者发送数据流；

多个处理这个数据流的数据流处理器，每一个数据流处理器与一个数据流输入和/或者数据流输出相连，并且包括

一个接收处理器，用于串行地处理从这个数据流输入所接收的数据流，和/或者

一个发送处理器，用于串行地处理需要被输出到数据流输出的数据流，

这个接收处理器将一个被处理的数据流写入到一个存储器，和/或者这个发送处理器从存储器中读取一个被处理的数据流，并且这个接收处理器可以被重新配置成从这个存储器中读取需要被处理的一个数据流，和/或者这个发送处理器可以被重新配置成将一个被处理的数据流写入到这个存储器。

10. 一个集成电路，包括：

多个串行数据流输入和/或者输出，它们串行接收和/或者发送数据流；

至少一组并行数据流输入和/或者输出，它们并行接收和/或者发送数据流；

多个处理这个数据流的串行数据流处理器，每一个串行数据流处理器与一个串行数据流输入和/或者数据流输出相连，并且包括

一个串行接收处理器，用于处理从这个串行数据流输入所接收的数据流，和/或者

一个串行发送处理器，用于处理需要被输出到数据流输出的数据流；和

至少被连接到这组并行数据流输入的一个并行数据流处理器，每一个并行数据流处理器包括

一个并行接收处理器，用于处理从这组并行数据流输入所接收的数据流，和/或者

一个并行发送处理器，用于处理需要被输出到这组并行数据流输入的数据流。

11. 一个集成电路，包括：

多个 I/O 管脚，接收和/或者发送信号；

一个数据流处理器，用于处理被这些信号所表示的数据；

一个可写入的配置区分符，用于规定其多个配置中的一个配置；和

配置电路，连接在多个 I/O 管脚和这个数据流处理器之间，并且对这个配置区分符作出响应以按照这个配置区分符所规定的来配置 I/O 管脚，

由此这个集成电路可以被用于多个发送协议。

12. 如权利要求 11 中所提出的这个集成电路，其中：

这个配置区分符规定了 I/O 管脚的电气特性；和

这个配置电路使用所需要的电气特性来配置 I/O 管脚。

13. 如权利要求 11 中所提出的这个集成电路，其中：

有多个数据流处理器；和

这个配置区分符规定了多个数据流处理器中哪一个数据流处理器被连接到这多个 I/O 管脚，

由此多个数据流处理器可以共享对被所接收信号和/或者多个 I/O 管脚所发送的信号所表示的数据的处理。

# 说　　明　　书

## 数字通信处理器

相关申请的交叉索引

这个专利申请要求下列美国临时申请的优先权:

§60/084,706，Brown，等人，可编程的包交换机，98 年 5 月 8 日申请；和

§ 60/105,823，Brown，等人，数字通信处理器，98 年 10 月 27 日申请。

本发明的领域

本发明一般涉及数字网络，更特别地，涉及在这样一个数字网络中所使用的交换机。

现有技术的描述：图 1

包和协议

在数字系统中所进行的通信一般是经过包进行的。图 1 中的 113 显示了一个包。简单地说，一个包是一个比特序列，这个比特序列的含义是由一个协议决定的。这个协议定义处理这个包的数字设备如何解释包中的比特。不管什么协议，大多数包具有一个报头 115，这个报头表示如何根据这个协议来处理该特定包，另外，包也具有一个负荷 117，这个负荷 117 是包所进行通信的实际信息。一个包也可能具有一个报尾 119，这个报尾 119 可能简单地仅指示包的结束，但是也可能包括允许对在包的传输或者处理期间所发生 的错误进行检测和/或者纠正的信息。取决于包协议的不同，这个包的长度可能是固定的，也可能是不定长的。在下面的讨论中，报头 115 和报尾 119 中的内容将被称作协议数据，因为解释这些内容的方式完全是由这个协议所决定的，而负荷 117 中的内容将被称作负荷数据。某个特定协议的包也经

常被称作帧或者信元。

在数字系统中，在许多不同的层上使用包来进行通信。这样，在这个数字系统中一个层上的一组包的负荷可能是一更高层的一个包。这被显示在图 1 的 137 中。IP 包是根据 IP 协议来被解释的一个包。IP 包 121 具有一个 IP 包报头 123 和一个长度可变的 IP 负荷 125。包括在 IP 报头 123 的信息中的是 IP 负荷 125 的长度。当 IP 包 121 被经过一个物理网络来传送时，IP 包被承载在一个传送包 127 流 135 的负荷中。每一个传送包 127 具有其自己的报头 129，负荷 131，和报尾 133。这里被称作传送包的是在 ISO-7 层模型中的链路层的包。传送包的长度可能是固定的或者也可能是可变的，这与链路层所使用的协议相关。

处理这个传送包的设备按照包的报头 129 和报尾 133 所指示的来对这个包进行处理，并且不检查负荷 131 内的内容。当一个传送包到达其目的地时，其负荷被传递到这个系统中、它所希望到达的部分，在这个情形下，其负荷到达根据 IP 协议而进行工作的一个部件，并且这个部件按照 IP 报头 123 中所指示的来处理 IP 包 121。当然，IP 负荷 125 仍可以是更高层上的另一个包。例如，它可以是需要发送到一个解密器的一个包，并且这个包的负荷可以是一个被加密的 IP 包 121。在这样一个情形下，处理 IP 包 121 的这个部件将这个负荷传递到这个解密器，这个解密器对加密的 IP 包 121 进行解密，并且将被解密的 IP 包返回到处理 IP 包的这个部件，以进行进一步的处理。当然，对 IP 包的这个处理可能包括将这个被解密的 IP 包发送到另一个目的地，并且如果与这个目的地的通信是经过传送层包 127 的协议来进行的，处理 IP 包的这个部件将向产生传送层包流的这个部件提供被解密的 IP 包，并且这个被解密的 IP 包将被携带在传送层包 127 的负荷中。

包交换机

当包被用于在相互远离的数字系统中进行通信时，包就在连接这个系统的数字网络中移动。在物理层，这个数字网络可以采用任何形式的媒质来在两个设备之间发送一个信号，例如，采用以太，一个导

电线，或者一根光纤。包被包交换机在传输路径之间进行路由。这个包交换机根据典型地被包括在包的报头中的信息来路由这个包。如人们所期望的是，每一类协议均有其自己的路由规则。例如，IP 协议使用逻辑路由；一个 IP 包的每一个源或者目的地均具有一个逻辑 IP 地址，并且用于发送到一个给定目的地的一个 IP 包在其报头中具有这个目的地的逻辑 IP 地址。这个报头不指出目的地的物理位置。这个 IP 包交换机必须将这个 IP 地址翻译成至少在到其目的地的部分路径上获得这个包的一个物理地址，并且也必须产生发送到这个物理地址的一个传送包流 135，在这个传送包流 135 的负荷 131 中携带这个 IP 包。这样，IP 节点 109（n ）位于以太局域网 105（a）的以太网节点 107（n）上，并且连接到这个 LAN105（a）的一个 IP 包交换机必须通过产生发送到以太网节点 107（n）的一个以太网包流来对被发送到 IP 节点 109（n ）的一个 IP 包作出响应，其中，在这个以太网包流的负荷中携带这个 IP 包。

在 101 处显示了一个典型的 IP 交换机。包交换机 101 被连接到一些物理媒质 106，通过这些物理媒质 106，这个包交换机 101 可以接收和发送数据。这样媒质的示例可以是光纤或者可以是由电导体组成的电缆。每一个这样的媒质 106 具有其自己的、用于定义经过这个媒质而发送的数据的协议；例如，用于经过一个光纤来发送数据的、被广泛使用的协议是 SONET 协议。在图 1 中，媒质 106( a..m )是使用 SONET 协议的光纤，而媒质 106（n..z）是电缆。在这个媒质层的包在这里被称作媒质层包，并且媒质层包的负荷是传送层包。就 ISO-7 层协议模型来说，这个媒质层包就是物理层的包。在交换机 103 中，经过光纤被发送和接收的传送层包是根据在 ATM 广域网 111 中被使用的 ATM 协议来产生的，而经过电缆被发送和接收的传送层包是根据在局域网 109 中被使用的以太网协议来产生的。在许多情形下，传送层包的负荷是 IP 包，并且在这些情形下，包交换机 103 将这个 IP 包路由到 IP 节点 109。如上面所描述的，这是通过确定 IP 包为到达其目的地将经过的媒质 106（i），并且然后，根据这个媒质所要求的协议产生一个包

流，来完成的，其中，媒质层包流将与这个媒质一起被使用的传送层包流作为负荷，并且接着传送层包流将 IP 包作为的负荷。这样，如果包交换机 103 从 WAN 111 接收到需要被发送到 IP 节点 109（n ）的一个 IP 包，并且 IP 节点 109（n ）位于以太局域网 105（a）的以太网节点 107（n）上，包交换机 103 就必须以媒质 106（n）所要求的形式产生一个包流，这个包流的负荷是需要发送到以太网节点 107（n）的一个以太网包流，并且接着，这个以太网包流的负荷是 IP 包。

这样，交换机 103 必须能够执行下述功能:

§ 读取具有输入媒质协议所要求的形式的包流，并且检索作为这个包流的负荷的传送层包，和检索作为传送层包的负荷的其它类型的包;

§ 将在 ATM WAN 111 上所接收的传送层包路由到在 ATM WAN 111 上的另一个目的地;

§ 将在一个以太网 LAN 105 上所接收的传送层包路由到连接在包交换机 103 的某个以太网 LAN 上的另一个目的地;

§ 对 IP 包，按照 IP 地址所要求的来路由这个包。

§ 产生具有输出媒质协议所要求的形式的向外发送包流，并且这个向外发送包流的负荷是传送层包，并且接着，传送层包的负荷包括其它类型的包。

这样的路由可能需要将一个类型的传送层包翻译到另一个类型的传送层包。例如，如果从 ATM WAN 111 进来一个 IP 包，并且这个包的目的地是以太局域网 105（a）的以太网节点 109（n），包交换机 103 必须从 ATM 传送层包的负荷中提取这个 IP 包，并且然后将这个 IP 包放置在将发送到以太网节点 107（n）的以太网包的负荷中。

另外，包交换机经常被用于执行安全功能，例如滤波，加密/解密，或者加扰/解扰。在这里，包交换机 103 被显示为在一个专用网络 104 和一个公用网络 102 的边界。每一个 IP 包 121 的报头包括这个包的源 IP 地址和目的 IP 地址，并且专用网络 104 的安全政策禁止从公用网络 102 来的、具有某些特定源地址的 IP 包对专用网络 104 进行访问，

并且也禁止从私用网络 104 来的、具有某些特定源地址的 IP 包对公用网络 102 进行访问。交换机 103 通过将每一个输入 IP 包的源地址与需要被禁止其访问的一个源地址列表进行比较，并且如果这个输入包在这个列表上，就丢弃这个输入 IP 包，来对每一个输入 IP 包进行滤波。交换机 103 也采用类似的方法来对向外发送的 IP 包进行滤波。就加密/解密来说，包交换机 103 可能会从公用网络 102 接收一个 IP 包，这个 IP 包的负荷是需要发送到专用网络 104 中一个 IP 地址的一个被加密 IP 包。在这样一个情形下，包交换机 103 将获取这个被加密 IP 包，并且对这个 IP 包进行解密，然后，将这个 IP 包发送到在专用网络 104 中的目的地。类似地，包交换机 103 可以接收一个 IP 包，并且可以对这个 IP 包进行加密，然后在经过公众网络 102 发送其它 IP 包以前，将这个 IP 包作为另一个 IP 包的负荷，其中这个 IP 包是需要经过公众网络 102 被发送到属于专用网络 104 的一个目的地。

包交换机所提出的问题

这种包交换机的设计向工程师们提出了很多问题。从前述讨论中可以很明显地看出，一个包交换机必须执行复杂的动作，例如对输入包流中的负荷进行定位并且产生向外发送的包流，在传送层或者更高层来对包流进行路由，翻译包中的信息，进行滤波，和加密/解密。包交换机执行这些动作时，其速度必须很快，并且其吞吐量必须很高。这个包交换机也必须能够处理很多类型的业务，例如从电子邮件的业务到数字 TV 的业务，到分组电话的业务，其中，在电子邮件中所需要的是在提交了一个电子邮件后的一个合理时间内（以小时来度量），这个电子邮件能够到达，在数字 TV 的业务中，包到达其目的地时，包之间的时间间隔必须是固定的，而在分组电话中，不仅对包之间的时间间隔有严格的限制，而且对一个包在网络中从其源到目的地所经历的总时间长度也有严格的限制。

在现代电子器件中，通过使用专用硬件，已经实现了高速度、高吞吐量，并且也实现了令人满意的时间限制，同时，也使用可编程的

处理器来对付复杂程度。基于专用目的硬件的器件典型地是快速的，但是其价格昂贵，不灵活，并且也不能够进行复杂的处理；基于可编程处理器的器件典型地是便宜，灵活，并且能够进行任何所需要的处理，但是其缺点是速度慢。

这样，高速度包交换机是基于专用目的的硬件。如人们所希望的，这样的包交换机的速度很快，其吞吐量很高，并且能够满足时间限制，但是它们的价格是昂贵的，它们也不灵活，并且不能够执行复杂的功能，例如滤波或者加密/解密。另外，每一个传送层协议需要有自己的特殊硬件，并且因为这个原因，在一个高速度交换机中所使用的传送层协议的改变就需要改变这个交换机的专用目的硬件。

低速度的包交换机是基于可编程处理器的。另外，如人们可以预见的，在开始时，这些交换机的价格可以相对比较便宜，并且能够执行任何所希望的复杂功能，在处理传送层协议或者其它层协议的改变时，仅需要对这个处理器进行重新编程。但是，基于可编程处理器的包交换机没有足够的速度，吞吐量，或者能力来满足使用专用目的硬件制造的包交换机所能够达到的时间限制。

所需要的是这样一个包交换机，这个包交换机是灵活的，低价格的，并且具有能力来执行基于可编程处理器的包交换机能够执行的复杂特征功能，同时，也能够满足时间限制，并且能够提供高速度和高的吞吐量。这里所描述的数字通信处理器的一个目的就是提供这样一个包交换机。

通过提供包括多个数据流处理器，一个流上下文（context）处理器，一个队列管理器，和一个缓冲器管理器的一个集成电路，本发明克服了包交换机的和处理数据流的一般设备的前述问题。

正在接收一个数据流的一个数据流处理器从这个数据流中提取控制信息，将这个控制信息提供给这个上下文处理器以按照这个数据流的上下文所要求的来进行解释，并且使用这个上下文处理器所提供的结果来处理这个数据流。在这个数据流需要被进行进一步发送的情况下，这个数据流处理器将输入数据流中的负荷提供到这个缓冲器管理

器，以将这个负荷保存在一个缓冲器中，并且向这个队列管理器发送一个排队命令。这个排队命令包括一个描述符，这个描述符包括标识这个缓冲器的至少一个标记和被正在发送这个数据流的数据流处理器所读取的一个队列的一个队列区分符。这个队列管理器将这个描述符排在合适的队列中。当进行发送的这个数据流处理器从这个队列中取出这个描述符时，这个数据流处理器使用这个标记来从缓冲器中拾取负荷，并且使用这个负荷来产生一个输出数据流，在产生这个输出数据流的过程中，按照这个输出数据流所要求的来添加控制信息。这个描述符被在这个数据流处理器上运行的程序所完全定义，所以，这个队列管理器提供了一个通用的机制，来在一个接收数据流处理器和一个发送数据流处理器之间以一个有序的方式传递信息。

一个数据流处理器包括一个接收处理器，一个发送处理器，一个控制数据处理器，局部存储器，和一个 DMA 引擎，这个 DMA 引擎在这个接收器处理器，这个发送处理器，和这个缓冲器管理器之间、以及在局部存储器和这个缓冲器管理器之间提供了 DMA 访问。属于每一个数据流处理器的局部存储器，以及属于这个缓冲器管理器和这个队列管理器的局部存储器，均是一单个全局地址空间内的一部分，并且可以被任何能够访问全局地址空间的设备所读取或者进行写入。当一个数据流处理器正在接收一个数据流时，这个接收处理器和这个控制数据处理器按照如下的方式来进行合作，以处理这个流：发送处理器接收这个流，从这个流中提取控制信息，并且将这个控制信息传递到这个控制数据处理器，并且将这个负荷用 DMA 的方式传递到这个缓冲器管理器。而这个发送处理器在这个流的下一部分进行工作，这个控制数据处理器使用这个上下文处理器来处理已经被进行 DMA 操作的这个部分的任何上下文信息，并且向这个队列管理器发送带关于被进行 DMA 操作的负荷的描述符的一个排队命令。称作数据作用域（data scope）的一个数据结构简化了在这个控制数据处理器和接收处理器之间的信息交换。在这个发送处理器和这个控制数据处理器之间所进行的交互基本上是相同的，但是不同的是，负荷的移动方向是

相反的。

这些数据流处理器包括串行数据流处理器和至少一个并行数据流处理器。这个并行数据流处理器可以被用于将这个集成电路连接到另一个相同类型的集成电路，连接到一个总线，或者连接到一个交换构造(switching fabric)。

这个串行数据流处理器是高度可配置的。配置是通过全局地址空间内的寄存器来完成的。每一个串行数据处理器可以接收和/或者发送独立的数据流，或者串行数据处理器组可以被组合在一起来合作处理一个数据流。一个串行数据处理器的 I/O 脚可以被配置成满足不同物理媒质的电气要求，并且也可以被配置成以使在一个组合内的所有串行数据处理器接收相同的输入。在一个接收处理器或者一个发送处理器内的各种器件可以按照处理正在被接收或者被发送的特定类型串行输入流所要求的来进行使能或者非使能，并且这个接收或者发送处理器也可以再循环它已经处理的一个数据流。

在细读下述的详细描述和图以后，该领域内的技术人员就可以理解属于本发明的其它目的和优点，其中：

图 1 是一个网络中的一个包交换机的一个框图；

图 2 是包括本发明的数字信号处理器的一个包交换机的一个高层框图；

图 3 是根据本发明的数字通信处理器的一个高层框图；

图 4 是在数字通信处理器中的一个信道处理器的一个高层框图；

图 5 是全局地址空间的一个图；

图 6 是一个信道处理器的局部存储器的一个图；

图 7 是在一个信道处理器中所接收的包的处理的一个流图；

图 8 是一个信道处理器所输出的包的处理的一个流图；

图 9 显示了接收和发送数据作用域；

图 10 是一个接收处理器 421 的一个详细框图；

图 11 是一个接收字节处理器的一个框图；

图 12 是一个发送处理器 427 的一个详细框图；

图 13 是一串信道处理器的一个框图；

图 14 是一串信道处理器的局部和共享存储器的一个框图；

图 15 是执行处理器 313 的一个框图；

图 16 是构造处理器（fabric processor）303 的一个框图；

图 17 是 Rx 和 Tx 构造数据处理器的一个详细框图；

图 18 显示了通过将两个 DCP 连接在一起而形成的一个交换机；

图 19 显示了通过将多个 DCP 连接到一个交换构造而形成的一个交换机；

图 20 显示了如何通过将一个 DCP 和另一个类型的数字交换逻辑连接到一个交换构造而形成一个交换机的；

图 21 显示了一个优选实施方式中的表存储器 207 的细节；

图 22 显示了 TLE 301 的内部结构；

图 23 显示了寄存器存储器 2205 和控制存储器 2215 的细节；

图 24 显示了 TLE 301 所执行的命令；

图 25 显示了一个接收信道处理器 307（i）和一个发送信道处理器 307（j）如何进行合作来交换一系列传送层包；

图 26 是将 RxSDP 421 组合在一个串中的一个示例；

图 27 是将 TxSDP 427 组合在一个串中的一个示例；

图 28 显示了一个环型总线消息；

图 29 显示了到 QME 305 的信道处理器接口；

图 30 显示了在一个优选实施方式中所采用的队列命令；

图 31 显示了在一个优选实施方式中的队列数据结构；

图 32 显示了在一个优选实施方式中的多播数据结构；

图 33 是在一个优选实施方式中的 MCL 3123 的一个细节；

图 34 显示了用于管理队列的各种结构；

图 35 显示了 QME 305 的扩展接口；

图 36 显示了调度程序扩展接口的细节；

图 37 显示了在调度程序接口上消息所使用的定时；

图 38 是缓冲器管理的一个逻辑视图；

图 39 显示了 BME 305 的命令细节；

图 40 显示了 BME 305 的硬件的细节；

图 41 显示了 SDRAM 229 的内容的细节；

图 42 显示了环型总线节点接口的细节；

图 43 显示了其上实现了全局总线 319 和负荷总线 317 的一个总线结构；

图 44 显示了在图 43 的总线结构上进行的长操作和短操作；

图 45 显示了全局总线 319 和负荷总线 317 的实现细节；

图 46 显示了可配置管脚逻辑 443 的各种结构的细节；和

图 47 显示了被用于配置管脚和接收和发送处理器的寄存器。

图中的标号有 3 个或者更多的数字：右边两个数字是在剩余数字所表示的图中的标号。这样，其标号为 203 的一个部件首先在图 2 中作为部件 203 出现。

下述详细描述将从一个数字包交换机的结构和操作的一个整体视图开始，这个数字包交换机包括根据本发明的数字通信处理器，并且其后是这个数字通信处理器的结构和操作的一个整体视图，并且然后提供这个数字通信处理器的部件的结构和操作的细节。

包括这个数字通信处理器的一个数字包交换机：图 2

图 2 是使用一个数字通信处理器集成电路 203 而制造的一个包交换机 201 的一个框图，其中这个数字通信处理器集成电路 203 实现了根据本发明的这个数字通信处理器。集成电路 201 具有用于下述外部设备的接口：

§ 总共 16 对串行输入 204 和串行输出 206 205（0…15），用于根据传送协议而被发送或者接收的包；

§ 到一个可选数字交换机设备的一个 32 比特输出和一个从这个可选数字交换机设备来的 32 比特输入（接口 221）；

§ 到一个可选主机处理器 227 的一个 PCI 总线接口 225；

§ 到一个 SDRAM 缓冲器存储器 229 的一个 128 比特宽的接口 228；

§ 到一个 SRAM 翻译表存储器 207 的一个 64 比特宽的接口；和

§ 到一个队列存储器 213 的一个 32 比特宽的接口。

更详细地讨论这些接口，可以对数字通信处理器 203 进行编程，以使在一单个 DCP203 中的串行输入和输出可以被用于很多不同的媒质和传送层协议。如果其中正在使用 DCP 203 的网络改变了，这个 DCP 203 可以被重新编程以处理新的网络结构。通过将几个串行输入或者输出连接到高速度协议的传输媒质，可以处理高速度协议。在一个优选实施方式中，这个媒质和这个传送层协议包括

§ 10Mb 以太网；

§ 100Mb 以太网；

§ 1Gb 以太网；

§ T1/E1 接口；

§ T3/E3 接口；

§ OC－3c 接口；和

§ OC－12c 接口。

DCP 203 从输入 204 中接收媒质包，并且从输出 206 输出媒质包。在从一个输入 204 接收一个媒质包的时刻和从一个输出 206 发送一个媒质包的时刻之间所发生的处理与这个 DCP 如何被编程有关。对 DCP 203 进行编程的方法包括下述 的：

§ 每一个输入具有一个接收处理器，每一个输出具有一个发送处理器；这些处理器可以被独立地进行编程，以处理不同类型的媒质包，传送层包，和是传送层包的负荷的包；

§ 输入和输出可以被组合在一起；

§ 对与一个包流相关的状态进行的操作是可以被编程的；例如地址翻译和纠错码处理；

§ 在 DCP 中的、包的源与目的地之间的关系是可以被编程的；和

§ 从一个包的源被传递到一个包的目的地的信息是可以被编程的，这与包信息在目的地被解释的方式相同。

对一个典型的包交换应用，DCP 203 被编程为按照如下的来进行

工作：当在一个输入 204 上接收到每一个媒质包时，DCP 203 将从这个媒质包的负荷中来的数据保存在缓冲器存储器 229 的一个缓冲器 231 中；这个被保存的数据在这里被称作一个协议数据单元或者 PDU；在很多情形下，这个 PDU 将是一个传送层包，这个传送层包是这个媒质包的负荷的一部分。当输出一个媒质包时，DCP 203 从一个缓冲器 231 中检索这个 PDU，在这个 PDU 中进行任何需要的改变（例如，改变传送层包中的路由信息，或者改变传送层包的类型），并且将这个协议数据添加到这个媒质包中。

DCP 203 使用翻译表存储器 207 中的翻译表 209 来路由传送层和更高层的包。一旦这个包被路由，DCP 203 将把一个描述符放置在队列存储器 213 与输出 206 相关的队列 215 的末尾，这个描述符表示包括这个包的 PDU 的缓冲器，并且这个输出 206 是将输出所述包的输出。一般，每一个队列 215 与一单个输出 206 相关，但是在一个输入 204 上接收的包可以被放置在任何队列 215 的末尾，这样允许在一个输入 204 上被接收的包可以经过很多输出 206 而输出。一个包也可以是多播包，即，在多于 1 个的队列 215 中进行排队。然后，DCP 203 从与一个输出 206 相关的一个队列的头部取出描述符 217，并且将这个描述符所标识的缓冲器 231 中的内容输出到这个队列的输出 206。

DCP 203 也从一个可选的数字交换构造接收包数据，并且向这个可选的数字交换构造提供这个包数据，如 221 所显示的。这个交换构造可以是另一个与包交换机 201 类似的包交换机，或者它也可以是任何其它能够路由数字数据流的设备。例如，包交换机 201 可以与其它包交换机一起连接到一个交叉交换机（crossbar switch），或者甚至可以连接到一个总线。从接口 221 所接收的或者输出到接口 221 的包数据的路由基本上与上面所描述的、在一个串行输入 204 上所接收的包的相同。最后，DCP 203 可以经过 PCI 总线 225 从可选的主机 227 接收包数据，并且向这个可选的主机 227 提供包数据。

外部控制接口包括 GPIO 接口 223 和 PCI 总线接口 225。GPIO 接口 223 是用于监视和控制外部系统部件，例如 LED，非挥发性存储

器，物理层串行传输和接收部件，和电源供电部件，的一个公用设施接口。PCI 总线接口 225 用于在 DCP 203 和一个主机处理器之间进行通信，并且这个主机处理器可能会控制交换系统 201，并且也可以执行更高层的操作，例如对在系统 201 中所接收的包内容进行访问检查。

操作详细示例

背景技术部分的示例将被用于更详细地显示包交换机 201 的部件如何被编程来路由包。为了示例的目的，串行对 205（i）的串行输入 204（i）正在接收一个 SONET 包流，这个 SONET 包流的负荷是一个 ATM 传送层包流。这个 ATM 传送层包流的负荷是需要被发送到 IP 节点 109（n）的一个 IP 包，这个 IP 节点 109（n）是在连接到以太网 LAN 105（a）的一个设备 107（n）上。以太网 LAN 105（a）被连接到串行对 205（j）的串行输出 206（j）。因为包交换机 201 正在被用于路由 IP 包，所以 DCP 203 已经被编程为在串行输入 204（i）上的输入传送层包中扫描包括 IP 包报头的负荷。当发现了一个 IP 包的报头时，DCP 203 就开始将这个 ATM 传送层包流的负荷路由到被一个缓冲器标记 233 所规定的缓冲器存储器 229 中的一个缓冲器 231。如果这个 IP 包比这个缓冲器长，就采用附加的缓冲器。

当这个 IP 包正在被传送到缓冲器存储器 229 时，DCP 203 处理这个 IP 包的报头中的信息，来决定如何路由这个 IP 包，并且然后路由这个 IP 包。使用翻译表存储器 227 中的翻译表来完成对这个报头信息的处理。在这个情形下，需要进行两个翻译：在这个 IP 包的报头中的 IP 目的地地址需要被翻译为具有这个目的地地址的 IP 节点所在的设备 107（n）的以太网地址，和设备 107（n）的以太网地址需要被翻译为队列存储器 213 中的队列的一个标识，其中串行输出 206（j）从队列存储器 213 的这个队列输出以太网包。这些翻译中的一个翻译表表目 211（i），从 IP 目的地地址（IPA）到以太网地址（ENA）的翻译被显示在翻译表 209（a）中。

DCP 203 使用从这个 IP 包的报头和这个翻译表 209 中获得的信息

来产生关于这个 IP 包的一个描述符 207. 包括在这个描述符 207 中的是以太网地址和包括这个包的缓冲器 231 的缓冲器标记 233. 然后, DCP 203 将这个描述符 207 放置在队列 215（j）的末尾 221，这个队列 215（j）是用于保存将从串行输出 206（j）输出的包的。当描述符 207 到达这个队列 215（j）的头部时，DCP 203 拾取包括传送层包的这个缓冲器 231 的内容，并且将这个内容变为具有适合于串行输出 206（j）的媒质协议的一个包流。这些媒质包的负荷是以太网传送层包。使用信息描述符 207，DCP 203 将设备 107（n）的以太网地址给予这个以太网传送层包。而这个以太网传送层包的负荷是被保存在这个缓冲器标记所规定的缓冲器中的 IP 包.

这里，应注意，DCP 203 当然对总共 16 个串行输入传送层包流和 16 个串行输出流同时执行上面所描述的操作或者这些操作的变形，并且在某些情形下，也同时在其自己和一个数字交换机之间通过接口 221，和/或者在其自己和外部主机 227 之间经过 PCI 接口 227 传送数据流。另外，如上面所描述的，在很多情形下，这个包交换操作是由严格的定时限制所控制的。如下面将更详细描述的，在 DCP 203 的设计中关键是提供 DCP 203 内部的数据路径和存储器结构，这些 DCP 203 内部的数据路径和存储器结构具有前面所描述的操作类型所需要的速度和潜在特性。

DCP 203 的结构：图 3 和图 5

图 3 是 DCP 203 的内部结构的一个高层框图。出现在图 3 中的、图 2 中的部件的标号与图 2 中对应的标号相同。从其上发送和接收传送层包的串行输入和输出 205 开始，每一个串行对 205 被连接到其自己的可编程信道处理器 307，这个可编程信道处理器 307 处理从这个对来的串行输入和到这个对的串行输出。这样，在这个优选实施方式中，有 16 个信道处理器 307. 对极高速度的传送层协议，最多可有 4 个信道处理器 307 可以被组合在一个信道处理器串中，如 309 所显示的.

构造处理器（fabric processor）303 与信道处理器类似，其不同点是这个构造处理器 303 处理从接口 221 所接收的和提供到接口 221 的并行数据。表搜寻引擎 301 使用翻译表存储器 207 中的地址翻译来进行地址翻译。队列管理引擎 305 管理描述符的队列 215。在某些实施方式中，这个队列被保存在 DCP IC 203 的存储器中；在其它实施方式中，队列存储器 213 是一个独立的外部存储器。缓冲器管理引擎 315 管理缓冲器存储器 229 中的缓冲器 231。执行处理器 313 对其它部件中的数据进行初始化并且维持这些部件中的数据，使用可选的外部主机 227 和 GPIO 接口来 PCI 总线接口，并且在需要的时候执行更高层的处理。执行处理器 313 的程序和数据被保存在 SDRAM 229 中。执行处理器 313，信道处理器 309，和构造处理器 303 均使用 TLE 301，QME 305 和 BME 315 的公用程序来处理包和/或者帧，并且在这里它们被统称作包处理器。但是，这里应指出的是，一个包处理器不仅可以被用于处理包，而且可以被用于处理任何其它的数据流，并且可以被认为是一个通用的比特/4 比特/字节/或者（在构造处理器的情形下）32－比特字流处理器。

DCP 203 的所有处理部件是可以被编程的。这个信道处理器 307 可以被独立地编程为处理不同类型的媒质层包，传送层包，和传送层包的负荷，并且构造处理器 303 可以被编程为处理在不同交换设备中所采用的数据。表存储器 207 中的表可以被写入，也可以被读取，并且表搜寻引擎 301 可以被编程为对表执行不同类型的搜寻。队列管理引擎 305 可以被编程为建立不同数目的队列，并且在队列中使用不同尺寸的描述符，缓冲器管理引擎 315 可以被编程为缓存不同尺寸的库，并且在库中具有不同的缓冲器大小。最后，XP 313 是一个通用处理器，并且可以被编程为执行任何功能。当 DCP 203 被初始化时，部件的程序被载入。程序代码可以被外部主机 227 载入到 SDRAM 229 中，或者这个程序代码也可以被保存在一个外部 PROM 中，这个外部 PROM 是由 BME 315 所管理的地址空间中的一部分。在任何一个情形下，XP 313 将这个代码载入到部件的存储器中。

数字通信处理器 203 的总线和存储器结构使 DCP 203 可以满足包交换的速度和时间限制，同时 DCP 203 可以采用表搜寻引擎 301，队列管理引擎 305，和缓冲器管理引擎 315 作为共享的资源。除了表搜寻引擎 301 外，数字通信处理器 203 的所有部件共享一单个全局地址空间 321。每一个包处理器的局部存储器在全局地址空间 321 中，并且每一个包处理器可以访问其局部存储器属于全局地址空间 321 的其它包处理器的局部存储器，也可以访问属于 BME 315 和 QME 305 的存储器。每一个包处理器能够对其自己的局部存储器进行直接访问，并且能够经过一个 32－比特全局总线 319 对其它部件的局部存储器进行访问。另外，构造处理器 303 具有其自己的、到队列管理引擎 305 的路径 304。

图 5 提供了全局地址空间 321 的一个概述图。首先是全局地址空间的部分 504，它是由信道处理器 307（0…15）的局部存储器 501 组成的。部分 504 进一步被划分为属于信道处理器串 309 中每一个的串存储器 503。一给定信道处理器 307（i）对其自己的局部存储器 501（i）的访问速度最快，对一起组成其串存储器 503 的、其串中的其它信道处理器的局部存储器的访问速度是第二快的，对全局地址空间 321 中其余部分的访问速度是最不快的。其局部存储器是全局地址空间 321 中一部分的其它部件是构造处理器 303，其局部存储器见 505，QME 305，其局部存储器见 507，BME 315，其局部存储器见 513，以及 XP 313，其局部存储器见 517。共享全局地址空间的这些处理器一般可以使用全局地址空间来进行处理器间的通信，例如，处理器可以在全局地址空间中建立信号标记来协调它们的操作。为了建立和使用这样的信号标记，在这个优选实施方式中的处理器具有一个测试－和－设置－比特指令。在全局地址空间中可以获得的其它信息包括 QME 局部存储器 507 中的队列状态信息 505，缓冲器管理引擎局部存储器 513 中的缓冲器状态信息 515，和 XP 局部存储器 517 中的全局配置寄存器 519 和系统接口配置信息 521。最后，QME 305 将关于一个包处理器所读取的队列的队列状态信息写入到这个包处理器的局部存储器中。

继续参考图 3，每一个部件的局部存储器通过负荷总线 317 连接到缓冲器管理引擎 315。这是一个 128－比特宽的总线，它以 4 个字节的突发来进行工作，可以在 SDRAM 229 和其它部件之间传送高达 64 字节的数据。经过负荷总线 317 所传送的数据包括下述的：

§ 执行处理器 313 所使用的程序和数据；

§ 执行处理器 313 使用来配置 DCP 203 的数据；

§ 在 SDRAM 229 和一个包处理器之间被传送的协议数据单元；

§ 缓冲器标记 233；和

§ 被包处理器进行排队和从队列中取出的描述符。

在 SDRAM 229 和局部存储器之间进行的传送是通过一个直接存储器访问（DMA）机制来完成的。实现这个传送的部件向这个 DMA 机制提供了一个 DMA 指令，然后，这个 DMA 机制执行这个传送而不需要这个部件的干预。这个结构允许一个部件并行地传送协议数据单元并且执行其它处理，这大大地增加了 DCP 203 的操作速度和吞吐率。

表搜寻引擎 301 和包处理器均经过环型总线 311 连接。环型总线 311 是 64－比特宽的总线，并且在它所连接的节点之间是时分复用的。在任何一给定时刻，这些部件中的每一个被分配的环型总线时隙数在 1 到 5 之间。每一个时隙可以携带一个 64－比特的消息。因为这个总线在其连接的节点之间是时分复用的，并且每一个节点具有一预定最大数目的时隙，所以可以在一固定的时间内，保证一个消息能够从一个节点传送到环型总线 311 上的另一个节点。在一个目前优选的实施方式中，执行处理器 313 使用环型总线消息来配置和读取表存储器 207 中的表，并且这些包处理器使用环型总线消息来向表搜寻引擎 301 提供关于翻译的信息，并且表搜寻引擎 301 使用环型总线消息来向这些包处理器提供翻译的结果。任何连接到环型总线 311 上的设备可以向任何连接到环型总线 311 上的其它设备发送环型总线消息，并且也可以从任何连接到环型总线 311 上的其它设备接收环型总线消息，所以在其它的实施方式中，例如，环型总线消息可以被用于协调组成一个

串 309 的执行处理器 307 的活动。

DCP 203 部件的合作示例

在图 3 所提供的细节上，继续参考图 1 和 2，在输入线 204（i）上正在接收 ATM 包流。输入线 204（i）属于信道处理器 307（i）。当这个 ATM 包到达时，信道处理器 307（i）就开始一系列的 DMA 传送，其中首先将这些包传送到信道处理器 307（i）的局部存储器，然后将这些包传送到被信道处理器 307（i）所拥有的一个缓冲器标记 233 所规定的一个缓冲器 231. 在这个传送在进行时，信道处理器 307（i）在 ATM 包的负荷中扫描 IP 包。当它发现了一个 IP 包的开始时，信道处理器 307（i）从这个 IP 包的报头中提取这个 IP 包的目的地地址，并且为这个表搜寻引擎 301 产生包括这个目的地址的一个消息。在这个情形下，这个目的地址规定了连接到以太网 LAN 105（a）的设备 107（n）上的 IP 节点 109（n）。以太网 LAN 105（a）接收从串行输出 206（j）输出的包. 然后，信道处理器 307（i）将这个消息放置到它自己在环型总线 311 上的一个时隙中. 当信道处理器 307（i）接收到对带设备 107（n）的以太网地址的这个消息的一个答复和被串行输出 206（j）所提供服务的队列号码 m 时，它就产生包括至少以太网地址和关于这个缓冲器的缓冲器标记 233 的一个描述符 217（k）。然后，信道处理器 307（i）经过负荷总线 317 向属于信道处理器 307（i）的一个邮箱写入一个排队命令. 这个排队命令包括至少描述符 217（k）和这个队列号码 m. 队列管理引擎 305 通过将这个描述符 217（k）放置在队列 215（m）的末尾，来对这个命令作出响应。

串行输出 206（j）属于信道处理器 307（j），并且队列管理引擎 305 向这个串行输出 206（j）提供队列 215（m）头部的描述符。它通过将一个指定队列 215（m）的出列命令经过负荷总线 317 写入到其邮箱来完成这个. 队列管理引擎 305 通过经过负荷总线 317 向信道处理器 307（j）提供在队列 215（m）头部 219 的描述符 217，来对这个出列命令作出响应.

最后，在队列 215（m）头部 219 的描述符 217 是描述符 217（k）。一旦信道处理器 307（j）具有描述符 217（k），它就使用描述符 217（k）中的缓冲器标记 233 来开始从包括 IP 包的缓冲器 231 到信道处理器 307（j）的局部存储器的一个 IP 包的 DMA 传送。当 IP 包到达时，信道处理器 307（j）产生发送到以太网设备 107（n）的一个以太网传送层包流，并且这个以太网传送层包流的负荷是 IP 包，并且将这个以太网传送层包流输出到串行输出 206（j）。在这些包中的以太网地址当然是从描述符 217（k）中获得的。

DCP 203 的总线和存储器结构的优点

从前述描述中可以很明显地看出，DCP 203 的总线和存储器结构允许一个包处理器并行地进行包报头处理，协议数据单元的转发，和排队与从队列中进行取出；另外，不同的总线不仅提供不同的路径，而且也提供满足在路径上执行操作的冗余性要求的路径。这样，对时间最敏感的操作是包报头中信息的翻译，因为只有获得了翻译的结果后，才能够进行路由。因为每一个包处理器在环型总线 311 上具有时隙，所以每一个包处理器能够在一个得到保证的时间内访问表搜寻引擎 301，并且所以，能够满足翻译的时间限制要求。

另一方面，在包处理器和 SDRAM 229 之间进行的协议数据单元的传送需要进行高带宽的突发传送，并且确切地说，是经过负荷总线 317 在局部存储器和 SDRAM 229 之间的 DMA 传送所提供的高带宽突发传送。从缓冲器管理引擎 315 和一个信道处理器 307（i）发起的缓冲器标记传送和了管理引擎 305 之间的描述符到一个信道处理器 307（i）的传送对时间的要求也是非常严格的，所以，它们也是在负荷总线 317 上被完成的。

对时间要求不严格的操作可以通过在全局地址空间 321 中的读取和写入来完成。这样进行读取和写入操作所需要的时间与它们在全局地址空间中所处的位置相关。对全局地址空间中一给定处理器本身局部部分存储器进行读取和写入所需要的时间是最少的，对属于给定处

理器串 309 的处理器进行读取和写入所需要的时间是第二少的，对不属于给定处理器串 309 的处理器进行读取和写入所需要的时间是最长的。

除了 TLE 301 外，DCP 203 的所有处理器均共享全局地址空间 321 的这个事实使处理器之间进行通信更容易实现。例如，执行处理器 313 可以简单地通过将数据写入到全局地址空间 321 中与它们相应的部分，来进行初始化和/或者重新配置其它部件，一个包处理器 307 可以获得关于正在被队列管理引擎 305 所管理的队列 215 的状态信息，关于正在被缓冲器管理引擎 315 所管理的缓冲器 231 的状态信息，或者在其串 309 中的其它包处理器的状态，这个状态信息的获得可以通过简单地从全局地址空间中、属于这些设备的部分中读取状态信息来完成，并且这些处理器可以通过全局地址空间中的信号标记来协调它们的动作行为。一个接收包处理器可以进一步在某些应用中使用全局地址空间，来将它所接收的协议数据单元直接写入到将输出这个协议数据单元的发送包处理器的局部存储器。最后，执行处理器 313 能够使用全局地址空间来决定执行处理器 313 与 其共享全局地址空间的每一个处理器的状态。

在全局地址空间操作非常频繁的地方，可以提供专用的硬件支持。例如，构造处理器 303 能够对队列管理引擎 305 的全局地址空间进行其独有的访问，这样，就能够对状态信息进行排队而不需要增加全局总线 319 的负荷。类似地，每一个包处理器在全局地址空间中与其相关的部分中的 QME 305 中具有关于其邮箱的状态比特，这些比特被直接连接到队列管理引擎 305 中，以使每一个包处理器能够确定其邮箱的状态，而不需要增加全局总线 319 的负荷。

包处理器的细节

下述部分首先将详细描述信道处理器 307（0…n），然后，描述构造处理器 303，最后描述执行处理器 313.

一个信道处理器 307（i）的整体视图：图 4 和 6

图 4 是一个信道处理器 307（i）的部件的一个框图。如从图 3 中可以看出的，信道处理器 307（i）经过串行输入 204（i）接收串行包输入，经过串行输出 206（i）提供串行包输出，并且被连接到环型总线 311，并且在环型总线 311 中具有时隙，同时也被连接到负荷总线 317 和全局总线 319。集合路径 433 和 435 和串路径 437 和 439 允许信道处理器（i）与在其串 309 中的其它信道处理器 307 进行组合，以处理极高速度的传输媒质。

在一个高层次上，信道处理器（i）有 3 个部件：信道处理器 RISC 核心（CPRC）401，这是控制其它部件操作的、并且能够访问全局地址空间 321 的一个通用目的处理器；串行数据处理器（SDP）420，它的处理涉及从串行输入 204（i）中接收包并且向串行输出 206（i）输出包；和 DMA 引擎 411，它经过全局总线 317 在信道处理器 307（i）和 BME 315 或者 QME 305 之间进行数据传送。SDP 420 和 CP RISC 核心 401 均经过环型总线接口 415 连接到环型总线 311。SDP 420 具有两个子部件：RxSDP 421，它处理输入包；和 TxSDP 427，它处理输出包。

继续参考 CPRC 401 的细节，CPRC 401 是采用众所周知的 MIPS1 指令集的一个子集的一个通用目的微处理器。它可以在环型总线 311 上发送和接收消息，并且与 SDP 420 共享两个寄存器文件。提取空间 417 被用于保存 SDP 420 从输入包中所提取出的协议数据，以被 CPRC 401 所使用，而融合空间 419 被用于保存 CPRC 401 提供到 SDP 420 的协议数据，以在产生输出包时使用。

CPRC 具有 4 个上下文(context)，即，独立的寄存器文件集。CPRC 401 可以在上下文之间进行切换，这或者是对一个程序中的一个 BREAK 命令作出响应，或者是对一个硬件中断作出响应。上下文被根据其号码而进行排序，上下文 0 具有最高的优先级，上下文 3 的优先级最低。每一个上下文有一个上下文项寄存器和一个上下文返回寄存器；当一个上下文被改变时，在当前上下文中的需要被执行的下一个指令的地址被保存在当前上下文的上下文返回寄存器中，并且使用

被保存在新上下文的上下文项寄存器中的地址中的指令来继续执行。

有5个系统事件能够促使上下文进行切换:

§ 主复位

§ 非屏蔽中断

§ 跟踪中断

§ 用户中断1

§ 用户中断2

对主复位事件和跟踪中断事件的上下文项寄存器的设置是系统所定义的; 对其它事件, 它们是可以被编程的。

指令存储器403包括被CPRC 401执行的代码. 它仅能够被CPRC 401和CP 307(i)所属于的串309中的其它CP 307的CPRC所读取和写入。在一个优选实施方式中, 代码被按照如下的方法载入到IMEM 403中: 首先, 执行处理器313经过全局总线319将这个代码载入到DMEM 407, 然后, CPRC 401将这个代码从DMEM 407传送到IMEM 403.

DMEM 405是信道处理器307(i)的局部数据存储器。它可以被用于CPRC 401的局部存储器, 也可以经过负荷总线413被用作对数据进行DMA传送。DMEM 405, 提取空间417, 和融合空间419均是全局地址空间321的部分, 并且可以被信道处理器307(i)的串309中的其它信道处理器307所访问, 也可以经过全局总线317被DCP 203中的其它处理器所访问. 除了DMEM 405外, 在一个优选实施方式中实现这个结构的存储器部件有请求FIFO 406, MUX 407, 全局总线接口413, 和负荷总线接口411. MUX 407复用RxSDP 421, TxSDP 427, 负荷总线接口411和请求FIFO 406到DMEM 405的访问。反过来, 请求FIFO 406允许CCP 203中连接到全局总线319的其它处理器可以访问DMEM 405, 允许CPRC 401访问DM 405和全局地址空间321, 允许在信道处理器CP 307(i)的串309中的信道处理器307中的其它CPRC 401访问DMEM 405. MUX 407, DMEM 405和负荷总线接口411一起组成了DMA引擎441, 这个DMA引擎441经过

负荷总线 317 在 SRAM 229 和 CPRC 401 和 SDP 420 之间执行 DMA 操作。

如上面的结构所隐含的，RxSDP 421，TxSDP 427，和负荷总线接口 411 对 DMEM 411 具有第一访问优先级，而全局总线接口 413，CPRC 401，和其它 CPRC 401 必须争夺剩余的访问权。这样，这个结构在一个方面，将第一优先级给予了在 SDP 420 和 DMEM 405 之间进行的协议数据单元的 DMA 传送，而在另一个方面，将第一优先级给予了在 DMEM 405 和 SDRAM 229 之间进行的协议数据单元的 DMA 传送。

继续参考串行数据处理器 420 的部件的细节，RxSDP 421 专用于处理输入包流。RxSDP 421 从输入流中提取包括协议数据的字段，并且将一个字段的内容或者经过环型总线接口 413 提供给环型总线 313，或者经过提取空间 417 提供给 CPRC 401。RxSDP 421 经过 DMA 传送将这个协议数据单元从包流提供到 DMEM 405。RxSDP 421 有 3 个子部件：管脚逻辑（pin logic）443，它接收表示传输媒质中包流的物理信号；成帧支持处理器 407，它对这个包流中的媒质层包和传送层包进行定位；和字节处理器 453，它从这个传送层包和它们的负荷中提取协议信息，同时经过路径 425 将这个传送层包传递到 DMEM 405。字节处理器 451 可以将所提取的协议信息放置在提取空间 417 中，和/或者经过将环型总线接口 415 将它放置在一个环型总线消息中。

TxSDP 427 专用于产生一个输出传送层包流，这个输出传送层包流承载 TxSDP 427 经过 DMA 传送从 DMEM 405 中获得的协议数据单元。为了完成这个，TxSDP 427 将 CPRC 401 已经放置在融合空间 419 中的协议数据融合入协议数据单元。TxSDP 427 的部件在功能上与 RxSDP 421 的部件的功能是对应的。这样，字节处理器 453 在传送层包和其负荷中操作协议数据，成帧支持处理器 449 提供媒质层包所需要的协议信息，管脚逻辑 445 将这个数据变为用来输出这个数据的物理媒质所要求的形式。

SDP 420 的其它令人感兴趣的特征是再循环路径 441 和集合路径

433 和 435。再循环路径 441 允许被保存在 DMEM 405 中的包被返回到 RxSDP 421，以被进行进一步的处理并且被输出到 DMEM 405。集合路径 433 允许一个串 309 中的所有 RxSDP 421 接收相同的输入数据，而集合路径 435 允许 TxSDP 427 接收数据，以将数据从 CP 307（i）所属于的这个串中的其它 CP 307 中的 TxSDP 进行输出。

信道处理器 307 的操作的示例：图 25

图 25 显示了一个接收信道处理器 307（i）如何与一个发送信道处理器 307（j）进行合作，来在一个输入 204（i）上接收包括一系列传送层包的一个媒质层包流，并且在属于发送信道处理器 307（j）的一个输出 206（j）输出包括一系列传送层包的一个媒质层包序列，这个传送层包的负荷是在信道处理器 307（i）中所接收的传送层包的负荷。所接收的和所发送的媒质层包和传送层包当然可以属于不同的协议。

所接收的媒质层包是在信道处理器 307（i）中的 RxSDP 421 中被接收的。RxSDP 421 从这个传送层包和传送层的负荷中提取协议数据，并将所提取的协议数据提供到提取空间 417，也经过 DMEM 405 和负荷总线 417 将由传送层包组成的协议数据单元 DMA 传送到 BME 315，BME 315 将这个协议数据单元放置在 SDRAM 229 的缓冲器 231 中，其中这个协议数据单元在 2503。同时，在信道处理器 307(i)中的 CPRC 401 使用这个协议数据来产生一个描述符 217，并且它经过负荷总线 317 将这个描述符 217 转发到 QME 305 以进行排队。（没有在这里显示的是，经过环型总线 311 将某些协议数据发送到 TLE 301，以进行翻译）。当 CPRC 401 发送需要被排队的描述符 217 时，CPRC 401 规定它应被排队在一个队列 215 的末尾，这个队列 215 的头部正在被发送信道处理器 307（j）所读取。QME 305 将描述符 207 排队在规定队列 215 的末尾。

当信道处理器 307（j）从队列 215 的头部中对一个描述符 207 进行队列取出时，QME 305 经过负荷总线 317 将描述符 207 发送到信道处理器 307（j）。信道处理器 307（j）使用描述符 207 来产生需要被

输出的包流的协议数据，并且将这个协议数据放置在融合空间 419 中。然后，信道处理器 307（j）发起一个 DMA 操作，经过负荷总线 317 和 DMEM 405 将这个协议数据单元 2503 从 SDRAM 229 中的缓冲器 231 传送到串行数据处理器 420. 在串行数据处理器 420 中，TxSDP 427 加上为输出 206（j）产生一个媒质层包流 2505 所需要的协议数据，这个媒质层包流承载了从 204（i）上所接收的包来的协议数据单元 2503。

局部存储器 501 的细节：图 6

图 6 显示了信道处理器 307（i）的局部存储器 501（i）。如前面所指出的，所有的局部存储器 501（i）均可以被共享全局地址空间 321 的数字通信处理器 203 的任何部件所读取和写入。

数据作用域 625（0）和 625（1）

如上面所描述的，RxSDP 421 将输入包流 DMA 传送到 SDRAM 229，并且当 RxSDP 421 这样做时，RxSDP 421 从输入包流中提取协议数据，并且将协议数据提供到 CPRC 401 以进行处理，而 TxSDP 427 从 SDRAM 229 接收输出协议数据单元，并且当 RxSDP 421 这样做时，TxSDP 427 从 CPRC 401 接收协议数据并且将协议数据放置在输出包流的合适位置中。对一给定传送层包，然后，这个处理包括两个步骤。在一个输入包的情形下，它们是：

§ 提取这个协议数据，并且将这个协议数据单元 DMA 传送到 SDRAM 229；和

§ 在 CPRC 401 处理这个协议数据。

在信道处理器 307 中，这两个步骤是流水线处理的：在输入包流的情形下，CPRC 401 处理从前一个协议数据单元中提取的这个协议数据，而 RxSDP 421 从这个包流中提取协议数据，并且对从这个包流中提取的当前协议数据单元进行 DMA 传送。发送也被进行类似的处理，其中 TxSDP 427 发送当前协议数据单元，而 CPRC 401 处理需要被包括在下一个协议数据单元中的这个协议数据。

通过使用数据作用域 625（0）和 625（1），就可能进行流水处理。数据作用域 625（0）和 625（1）是可以被 CPRC 401 所看见的和所访问的数据结构，并且它们也控制 SDP 420 的操作和控制在 CPRC 401 和 SDP 420 之间进行的交互作用。一给定数据作用域 625（i）包括一组数据作用域寄存器 624 和在数据作用域事件寄存器 632 中的一组标记 632（i）。数据作用域寄存器 624 进一步由一个 Tx 数据作用域 641 和一个 Rx 数据作用域 643 组成。Rx 数据作用域 643 在第一步骤期间，接收 RxSDP 421 从输入包流中提取的协议数据，并且在第二步骤中 CPRC 401 处理这个协议数据；类似地，Tx 数据作用域 641 在第一步骤期间，接收 CPRC 401 为输出包流而处理的这个协议数据，并且在第二步骤期间，TxSDP 427 输出来自 Tx 数据作用域 641 的这个协议数据。

除了提供包处理的流水操作外，数据作用域 625 也为在 CPRC 401 上被执行的程序提供了到当前正在被 SDP 的 RxSDP 421 所接收的、或者正在被 TxSDP 427 所输出的包流的一个统一接口。在需要对一个流进行进一步处理的应用中，可以增加数据作用域的数目。例如，包括处理一个输入流，将所产生的 PDU 保存在 DMEM 405 中，然后使用 再循环路径 441 又处理被保存在 DMEM 405 中的 PDU，和然后将最后的 PDU 使用 DMA 方式传送到 SDRAM 229 的 RxSDP 的处理可能包括 4 个数据作用域。

在 CPRC 401 上执行的程序决定哪一个数据作用域目前正在被 SDP 420 所使用。在 SDP 420 将协议数据提取到数据作用域 625（0）并且从数据作用域 625（0）融合协议数据时，CPRC 401 处理在数据作用域 625（1）中的数据。当 SDP 420 完成了对数据作用域 625（0）的处理时，它向 CPRC 401 发送信号，并且 CPRC 401 建立数据作用域 625（1），以使 SDP 420 能够在数据作用域 625（1）上进行工作，并且 CPRC 401 自己开始在数据作用域 625（0）上进行工作。

继续更详细地参考数据作用域 625（i）中的内容，Rx 数据作用域 643 包括提取寄存器 601，提取寄存器 601 包括被 RxSDP 601 所提取

的协议信息；RxCB 633，包括将 RxSDP 421 所接收的包 DMA 传送到 SDRAM 229 所需要的信息；和 Rx 状态 635，包含关于 RxSDP 421 的状态信息，包括 RxSDP 421 是否完成了它正在进行的 DMA 传送。Tx 数据作用域 641 包括用于包发送的模拟寄存器。融合寄存器 603 包括需要与输出包融合的协议数据，TxCB 633 包括从 SDRAM 229 对 TxSDP 所正在发送的包进行 DMA 传送所需要的信息，并且 Tx 状态包括关于 TxSDP 427 的状态信息，包括 TxSDP 427 是否完成了它正在进行的 DMA 传送。

控制块寄存器 611

控制块寄存器 611 是控制在 CPRC 401 和 SDRAM 229 之间进行的 DMA 传送的一组 4 寄存器。一个 WrCB 610 控制从 CPRC 401 到 SDRAM 229 的 DMA 传送，而一个 RdCB 控制到 CPRC 401 的 DMA 传送。

环型总线控制寄存器 617

这些寄存器是环型总线接口 415 的一部分。它们允许 RxSDP 421 和 CPRC 401 在这个环型总线上发送消息，并且允许 CPRC 401 在这个环型总线上接收消息。发送消息使用 4 个寄存器，接收对被 CP 307（i）所发送消息的答复使用 8 个寄存器，接收一个未经请求的消息使用一个寄存器队列。

SONET 开销比特 612

这个寄存器包括 CP 307（i）所输出的 SONET 包的 SONET 开销比特。

RxSDP 控制 613 和 TxSDP 控制 615

这些寄存器包括分别控制 RxSDP 421 和 TxSDP 427 的操作的参数。

CP 模式寄存器 625

这个寄存器包括控制 CP 307（i）的操作的参数。

SDP 模式寄存器 627

这个寄存器包括控制 SDP 420 的操作的参数。

队列状态 621

队列状态 621 包括关于 QME 305 中信道处理器 307（i）的邮箱状态的信息，和关于正在被信道处理器 307（i）所读取的队列的状态信息。这个寄存器中用于指示信道处理器 307（i）的邮箱状态的硬件直接被 QME 305 所控制。这样，读取和写入这个寄存器不会对全局总线 319 产生任何负荷。QME 305 经过负荷总线 317 将正在被信道处理器 307（i）所读取的队列的状态信息 DMA 传送到 DMEM 405。

事件定时器 620

这个寄存器包括可以被在 CPRC 401 中所执行的软件设置和启动的一个事件定时器；当这个定时器溢出时，有 CPRC 401 的事件机制对其作出响应的一个事件结果。

循环计数器寄存器 619

循环计数器寄存器 619 包括一个计数器值，一个时钟分频值，和一个 CC 使能比特。CPRC 401 可以设置这个计数器值，这个时钟分频值，和这个 CC 使能比特。这个时钟分频值规定了计数器值相对 CPRC 401 的时钟进行加 1 操作的速率。当 CPRC 401 设置了 CC 使能比特时，这个计数器就开始运行；当这个 CPRC 401 清除了这个 CC 使能比特时，这个计数器就停止运行。当前的计数器值不会受到设置或者清除 CC 使能比特的影响。

事件寄存器 631

这个寄存器包括表示是否出现了这个 CPRC 401 必须对其作出响应的一个异步事件的标记。有两类事件：普通事件，其标记在寄存器 630 中，和与数据作用域 625 相关的数据，其标记在数据作用域事件寄存器 632 中。

除了这里描述的所有寄存器，一个 CPRC 401 也可以访问在其局部数据存储器 405 中的数据，并且能够如上面所描述的在 SDRAM 229 和局部数据存储器 405 之间、经过局部存储器 405 在一个 SDP 和 SDRAM 229 之间进行 DMA 传送。

CPRC 401，RxSDP 421 和 TxSDP 427 进行合作的示例的细节：图 7－9

图 7 提供了 RxSDP 421 和 CPRC 401 如何被编程来在处理一个输入包流中进行交互通信的一个概述图。流图 701 不仅指示被执行的操作，而且还指出 RxSDP 421，CPRC 401 和 QME 305 中的哪一个执行这些动作。在 703 开始，RxSDP 421 读取一个输入包（705）. RxSDP 421 对它所读取的东西所进行的处理与这个材料在包中的位置相关。有 3 类信息是 RxSDP 421 必须处理的：协议数据单元，必须被 TLE 301 所翻译的协议信息，和必须被 CPRC 401 所处理的协议信息。如在块 707 所显示的，当 RxSDP 421 读取协议数据单元时，这个 RxSDP 421 经过 DMEM 405 将协议数据单元用 DMA 方式传送到 SDRAM 229；如在 709 所显示的，RxSDP 421 使用环型总线 311 向 TLE 301 发送带 TLE 301 必须翻译的协议信息的一个消息；如在 711 所显示的，最后，RxSDP 421 使用提取空间 601 将 CPRC 401 需要处理协议信息的信息传送到 CPRC 401。

在模块 713，CPRC 401 使用它从 RxSDP 421 所接收的信息和它从 TLE 301 所接收的、对 RxSDP 421 的消息作出响应的答复，来决定需要对这个传送层包进行什么处理。如果这个传送层包是无效的，或者因为其内容被破坏或者因为其负荷是从被包交换机所滤波的一个

源来的，CPRC 401 将这个包标记为可以被丢弃。对这个标记作出响应（标记被包括在在 DMA 期间被添加的一个末尾中），这个 DMA 引擎停止进行发送，并且 BME 315 释放正在接收这个包的缓冲器 231。

如果这个传送层包是有效的，CPRC 401 使用它已经从 TLE 301 所接收的信息和在提取空间 601 中的信息来决定来自这个包的协议数据单元需要被放置在其中的队列 215，并且产生关于这个队列的一个描述符 217。然后，在 751 中，CPRC 401 发送一个排队命令到负荷总线 317 上，这个排队命令包括这个描述符和这个协议数据单元需要被放置在其中的队列的号码。

QME 305 通过将这个描述符 217 放置在合适的队列 215 中，来对这个排队命令作出响应。如在块 717，719 和 721 所显示的，取决于读取这个队列的 DCP 203 的部件，有 3 大类队列：被 XP 处理器 313 所读取的队列；被一个信道处理器 307 所读取的队列；和被构造处理器 303 所读取的队列。在队列被 XP 处理器 313 所读取的情形下，与这个描述符相应的这个协议数据单元可能会发送到主机 227；在队列被构造处理器 303 所读取的情形下，与这个描述符相应的这个协议数据单元可能会发送到一个交换构造。

这里应指出的是，因为 RxSDP 421，CPRC 401 和 QME 305 均是独立操作的处理器，流图 701 中所显示的处理可以被并行进行。RxSDP 421，和 CPRC 401 的流水线处理已经被描述了；另外，CPRC 401 不必要等待 QME 305 对 CPRC 401 的排队命令作出响应。

继续更详细地参考数据作用域 625 在 RxSDP 421 和 CPRC 401 之间的交互通信中的作用，图 9 显示了接收数据作用域 643 的细节。从 Rx 状态寄存器 635 开始，这个寄存器指出在 RxSDP 421 和 CPRC 401 之间的交互通信的状态；这个寄存器具有 4 个感兴趣的字段：OWN 字段 935 被硬件设置；这个比特指出 RxSDP 421 或者 CPRC 401 目前是否正在使用 Rx 状态 635 目前所属于的数据作用域 625。L5: L0 937 是在程序控制下被 RxSDP 421 和 CPRC 401 所设置和复位的 6 个握手比特。忙 941 被 RxSDP 941 所设置，并且指出 RxSDP 421 是否是忙

的。Tx 状态 639 在功能和内容上与 Rx 状态 635 类似。

RxCB 633 控制在输入包的 DMA 期间，在 RxSDP 421 和 DMA 引擎 441 之间的交互通信。当 CPRC 401 拥有 RxCB 633 所属于的数据作用域 625（i）时，CPRC 401 建立 RxCB 633，并且当 RxSDP 421 和 CPRC 401 交替拥有数据作用域 625（i）时，RxSDP 421 使用 RxCB 633 来继续对输入包进行 DMA 传送. RxCB 633 中的大多数字段包括执行 DMA 处理所需要的各种寻址信息。缓冲器池号码 909，BTAG 933，和偏移 931 一起规定了 DRAM 229 中、DMA 引擎 441 当前正在写入 RxSDP 421 所接收的包的位置。

如下面将更详细描述的，DRAM 229 被划分为缓冲器池. BTAG 933 是这个池中的缓冲器的缓冲器标记 233，偏移 931 是缓冲器中、数据当前正在被写入的偏移。当 DMA 引擎 441 写入数据时，它更新偏移 931. DMEM DMA 地址 907 是 DMEM 405 中的 16 字节数据线的地址，其中 DMA 引擎 441 当前正在将数据从 DMEM 405 进行 DMA 传送到 DRAM 229. Txrcy 地址 905 和 Rxrcy 地址 903 就是当 RxSDP 421 正在从 DMEM 405 重新循环数据时，RxSDP 421 所使用的特殊地址. Txrcy 地址 905 规定了 TxSDP 427 的 DMA 引擎 441 当前正在将数据写入到其上的 DMEM 405 线，而 Rxrcy 地址 903 规定了 RxSDP 421 正在向其写入数据的 DMEM 405 线。这样，这些地址允许 RxSDP 421 重新循环包，或者是在这些包被写入到 SDRAM 229 以前，或者是在这些包被写入到 SDRAM 229 以后. DMEM 线地址 901 是 RxSDP 421 正在为其写入数据的 DMEM 405 线。

RxDBCTL 913 包括控制在 CPRC 401，RxSDP 421 和 DMA 引擎 441 之间的交互通信的控制和状态字段：

§ Avail 929 指出 RxCB 633 是否可以被使用；

§ NR 927 指出在放弃传送以前，为了将数据传送到 DRAM 229，DMA 引擎 441 应产生的请求次数；

§ Error 925 指出在 RxCB 933 目前所代表的传送期间，是否出现了一个错误；

§ Own 921 指出 RxSDP 421 向其写入、并且 DMA 引擎 441 从其进行读取的 DMEM 405 中的线目前是否正在被 RxSDP 421 写入或者是否被 DMA 引擎 441 进行读取;

§ EOP 917 被 RxSDP 421 设置，当 RxSDP 421 在将被写入到 DMEM 405 中的线的数据中遇到一个包结束指示时;

§ ST 915 是 SDP 420 的目前状态;

§ BCTL 状态 919 是负荷总线 317 的目前状态; 和

§ 长度 911 被 RxSDP 421 设置。它是 RxSDP 421 正向其进行写入的 DMEM 405 的线中的数据的长度。

TxCB 637 基本上与 RxCB 633 类似，除了它所控制的 DMA 传送在相反方向进行，并且这些字段具有与这个方向相应的意义。

当 RxSDP 421 设置 OWN 比特 935，在 Rx 状态寄存器 636 中的 L2 Done 937 或者 L1 Done 939，或者在 RxCB 633 中的 Avail 929 时，这个结果是对 CPRC 401 的一个中断; 哪一个动作将产生这个中断由数据作用域的数据作用域事件寄存器 632 中的多个比特所指示，该数据作用域是当 RxSDP 设置所讨论的比特时所在的作用域。相同的结构被用于 Tx 数据作用域 641 中的相应比特。

开始时，CPRC 401 已经建立数据作用域 625（0），并且将对数据作用域 625（0）的拥有关系给予 RxSDP 421。现在，CPRC 401 拥有数据作用域 625（1）。当一个包进入时，RxSDP 421 提取协议数据并且将这个协议数据写入到提取空间寄存器 601（0）。RxSDP 421 在 Txmsg 645（0）中将一个消息发送到 TLE 301，该消息中包括需要被翻译的协议数据。查找的结果将出现在一个 RxResp 寄存器 647（0，i）中。当这个在进行时，RxSDP 421 开始将这个协议数据单元写入到在 RxCB 633（0）中所规定的 DMEM 405 中的线。当已经接收了整个线时，RxSDP 421 设置 Rx 状态 635 中的拥有者比特 935，来指出 CPRC 401 现在拥有数据作用域 625（0），设置 RxCB 633（0）中的拥有者比特 921，来指出 DMA 引擎 441 现在可以读取它已经写入的线，并且产生自动地将 RxSDP 421 切换到数据作用域 910（1）的一个信号。

然后，RxSDP 421 检查数据作用域 625（1）的拥有者比特 935，来看 CPRC 401 是否已经控制了它。如果 CPRC 401 已经控制数据作用域 625（1），在 RxSDP 421 处理这下一个包以前，RxSDP 421 就进行等待，直到 CPRC 401 放弃对数据作用域 625（1）的控制。处理如上所述，除了 RxSDP 421 使用数据作用域 625（1）的资源，而不是数据作用域 625（0）的资源外。

当 RxSDP 421 使用数据作用域 625（1）来进行工作时，CPRC 401 处理接收数据作用域 625（0）。CPRC 401 检查和/或者删除 RxSDP 421 放置在提取空间寄存器 601（0）中的这个协议数据，检查 RxCB 633（0）来决定到 SDRAM 229 的 DMA 是否已经无错误地完成了，并且建立 RxCB 633（0）以被 RxSDP 421 下一次使用。然后，CPRC 401 使用 RxSDP 421 放置在提取寄存器 601 中的协议数据和在 RxRsp 结构 647（0，i）中从 TLE 301 所接收的任何翻译，来为正在被写入到 SDRAM 229 的数据产生一个描述符，并且将这个描述符放置在 QME 305 中的信道处理器 307 的邮箱中。当 CPRC 401 已经做完了所有这些时，它设置拥有者比特 935（0），以使接收数据作用域 625（0）又可以被 RxSDP 421 所使用。如果设置了拥有者比特 935（1），表示 RxSDP 421 完成 了使用接收数据作用域 625（1）进行的工作，CPRC 401 使用与上面对接收数据作用域 625（0）所描述的方式相同的方式，在接收数据作用域 625（1）进行工作。

图 8 提供了 CPRC 401 和 TxSDP 429 如何在发送一个传送层协议流中进行交互通信的一个概述图。发送比接收简单得多，因为输出包仅需要被装配和发送，不需要被解释。在发送时所需要做的大多数工作是被 CPRC 401 完成的。从 803 开始，CPRC 401 首先执行循环 805；即，CPRC 401 检查它从 QME 305 所接收的队列状态信息，以看在正在被信道 处理器 307（i）所读取的一个队列 217 中是否有一个描述符。如在 807 所显示的，如果有一个描述符 217，CPRC 401 将一个出列命令放置在负荷总线 317 上，并且也经过负荷总线 317 从 QME 305 接收这个描述符。然后，CPRC 401 使用在这个描述符中的信息来建立

正在被发送的包所需要的融合寄存器 603（811），使用这个描述符中的缓冲器标记来在 Tx 控制块寄存器 637 中建立寄存器以将这个缓冲器内的内容从 SDRAM 229 传送到 TxSDP 427（813），并且然后，开始这个传送（815）。当状态 915 或者 EOP 917 指示这个传送结束时，CPRC 401 释放 TxSDP 427 和在这个传送中所涉及的其它资源（817）。TxSDP 427 和 CPRC 401 以与 RxSDP 421 和 CPRC 401 进行交替的方式相同的方式来交替使用数据作用域 625（0）和（1）。因为在数据作用域之间的交替是在 CPRC 401 所执行程序的控制下进行的，所以这个程序可以做任何所需要的处理，来在 TxSDP 427 和 RxSDP 421 之间分配一个数据作用域所表示的资源。

RxSDP 421 的细节：图 10 和 11

RxSDP 421 和 TxSDP 427 向每一个信道 处理器 307（i）提供了在串行输入 204（i），串行输出 206（i），和信道处理器 307（i）的其它部件之间的一个可编程接口。这样，一给定串行输入 204（i）可以被按照正确处理属于给定协议的媒质层和传送层包所需要的方式来进行编程，而一给串行输出 204（j）可以被编程成输出属于给定协议的媒质层和传送层包。每一个 SDP 421 或者 427 具有其自己的微程序存储器和独立的寄存器集合。

图 10 是 RxSDP 421 的一个框图。这些部件是将这个串行输入转换为并行，并且首先在媒质层包的级别上处理这个输入，然后在传送层包的级别上或者更高级别上来处理这个输入包的一系列处理器和 FIFOS。某些处理器是专用于处理某些特定协议的。一个旁路路径可以被编程为允许数据旁路任何处理器和/或者 FIFO。整体来说，这些部件如下所述：

§ 可配置的物理逻辑 443，它从物理层接收串行输入，并且对它进行解释以产生一个 10 比特代码的流。

§ 管脚接口 204（i），它从这个物理层接收 10 比特代码。

§ 8b/10b 解码 1001，它从管脚逻辑 443 所接收 的 10 比特代码产

生字节；

§小的 FIFO 1003，它是具有不同的、可编程的写入和读取时钟的一个异步 FIFO（队列）。这个写入时钟的运行频率是这个输入数据流所需要的一个频率，而这个读取时钟的运行速率是 CPRC 401 的速度。在一个目前的优选实施方式中，FIFO 1003 是 8 个 9 比特字深的 FIFO。

§接收比特处理器 1005，它对它从小的 FIFO 1003 中 所接收的字节流进行图形匹配和字段提取，被提取的字段经过路径 1005 被发送到提取空间 417。

§接收 SONET 帧模块 1007，它处理 SONET 帧。接收 SONET 帧模块 1007 对帧中的数据进行解扰，从帧中删除协议数据，进行奇偶校验，并且经过路径 1008 将这个协议数据写入到提取空间 417 中。

§接收同步处理器 1009，它处理 ATM 信元。接收同步处理器 1009 发现这个信元的边界，删除这个协议数据，使用这个协议数据启动一个 TLE 操作，对这个负荷进行解扰，并且经过路径 1019 将这个协议数据写入到提取空间 417 中。

§大的接收异步 FIFO 1011，它是一个异步 FIFO。在一个当前的优选实施方式中，FIFO 1011 是一个 64 个 10－比特字深的 FIFO。FIFO 1011 主要是用于当 TLE 301 查找 VPI/VCI 时，对一个 ATM 信元进行分级，并且 FIFO 1011 也用于使用高速度接收包来提供灵活性。

§接收字节处理器 1013，在微代码的控制下进行工作，来对 9－比特字的数据进行图形匹配和字段提取。

通过复用器 1002，1006，1014，和 1010 来实现旁路路径 1015。复用器 1010 也实现了再循环路径 441。旁路和再循环路径是可以通过设置 SDP 模式 627 中的比特进行动态地配置的。通过 RxSDP 控制 613 中的寄存器，数据可以直接在 CPRC 401 和比特处理器 1005，同步处理器 1009，或者字节处理器 1013 之间进行传送。

如前面所描述的，信道处理器 307 可以被进行组合来处理非常高速度的数据流。当被组合在一起时，被组合的信道处理器作为一个流

水线来进行工作，其中每一个被组合的信道处理器依次处理数据流中的一部分数据。一个集合中 RxSDP 421 之间的协调可以通过令牌总线 1027，1017，和 1028 来实现。例如，在一个组合中，一个被使能的 Rx 比特处理器 1005 仅当被令牌总线 1004 提供了这个令牌时，才处理输入数据流。几个令牌总线是需要的，因为用于处理一个输入流的 RxSDP 421 部件将随输入流的类型而变化。

可配置管脚逻辑 443 的细节：图 46 和 47

可以使用两个方法来配置配置管脚逻辑 443:

§ 向一个串中的一个，两个，或者 4 个 SDP 420 提供输入流，或者从一个串中的一个，两个，或者 4 个 SDP 420 中接收一个输出流；和

§ 使用这个用来发送输入流或者输出流的媒质所需要的不同物理接口来进行工作。

第一个类型的配置是为了允许将信道处理器 307 组合成一个串 307 以处理非常高速度的输入流或者输出流；第二个类型的配置允许 DCP 203 被用于不同类型的传送媒质，而不需要在 DCP 203 的外部增加设备以对从这个媒质所接收的信号进行适配，以被 DCP 203 所使用。在信道处理器的局部存储器中的一个寄存器控制这两个类型的配置。这些寄存器可以被这个信道处理器自己所设置，或者可以被 XP 313 所设置。

图 46 是一个表 4601，显示了在一个 DCP 203 中的一个串 309 如何被配置成来根据 RMII，OC－3，DS1，DS3，GMII，TBI，和 OC－12 标准来接收和产生串行数据流。列 4603 列出了这个串中的每一个信道处理器的 I/O 管脚；列 4605 指出这个管脚的一般目的；这样，在每一个信道处理器中，I/O 管脚 0 和 1 是用于时钟信号的，而剩余的管脚是用于数据的。剩余的列显示了这些管脚是如何被用于每一个媒质的：列 4607 指出在 RMII 中它们是如何被使用的；4609 指出在 OC－3 中它们是如何被使用的；4611 指出在 DS3 中它们是如何被使

用的；4615 指出在 GMII 中，当这个串中的两个信道处理器被用于接收数据，而两个信道处理器被用于发送数据时，它们是如何被使用的；4617 指出在 TBI 中，当这些信道处理器以相同的方式被使用时，它们是如何被使用的；和 4619 指出在 OC－12 中它们是如何被使用的，其中在一个串中的信道处理器交替作为发送器和接收器。

各种媒质需要不同类型的、用于 I/O 管脚的驱动器和接收器；这样，在可配置管脚逻辑 443 中的每一个 I/O 管脚具有一个 3－态驱动器，一个 TTL 驱动器，和一个 PECL 驱动器。对例如使用 PECL 逻辑的 OC－3 媒质，I/O 管脚对被配置为差分对，如列 4609 所显示的。

图 47 显示了管脚模式寄存器 4701 和 SDP 模式寄存器 4713。每一个信道处理器 307 具有这些寄存器中的一个。寄存器 4701 决定如何配置这个信道处理器的 I/O 管脚。数据 Cnfg 比特 4703 是决定 I/O 管脚是否被一个 3－态驱动器，一个 TTL 驱动器，或者一个 PECL 驱动器所驱动的 4 个比特。RxClk 复用器 4705 和 TxClk 复用器 4707 分别规定正在被用于携带接收和/或者发送时钟信号的管脚。Rx 数据使能 4709 规定了哪一个管脚被用于接收数据。最后，Tx 数据使能 4711 规定哪一个管脚被用于发送数据。

SDP 模式寄存器 4713 包括用于控制 RxSDP 421 中的哪一个部件被使能，循环电路中的什么被使能，和信道处理器 309 所属的串中目前采用哪个类型的组合的比特。信道处理器 TxSDP 427 也具有一个类似的寄存器。RxEn 比特 4715 指出是否使能 了这个信道处理器的 RxSDP 421；比特 4717 指出是否使能了信道处理器的字节处理器 313；比特 4719 指出是否使能了信道处理器的比特处理器 1005；比特 4721 指出是否使能了 Rx Sonet 帧模块 1007；比特 4723 指出是否使能了 Tx 同步处理器 1009。下两个比特是用于再循环控制的，其中比特 4725 指出到字节处理器 1013 的再循环和比特 4729 指出从提取空间 417 到比特处理器 1005 的再循环。组合模式字段 4731 是一个 2－比特字段，规定在这个串中是否没有进行组合，双向组合（即，两个信道处理器进行接收，两个信道处理器进行发送）或者 4 向的组合（所有 4 个信

道处理器或者进行接收或者进行发送）。

SDP 中处理器的实现方式：图 11

图 11 显示了 SDP 中的处理器是如何被实现的。更具体地说，这里所显示的处理器是 RxByte 处理器 1013，但是在 Rxbit 处理器 1005 中的处理器和接收同步处理器 1009 是类似的。SONET 帧模块 1007 是使用与图 11 中的一个处理器类似的一个处理器而实现的一个可配置状态机。

如在复用器 1107 所显示的，RxByte 处理器 1013 从大的 FIFO 1011 接收外部输入。RxByte 处理器 1013 可以经过复用器 1123 向环型总线接口 415，提取空间 417，或者缓冲器 1025 提供外部输出，而环型总线接口 415，提取空间 417，或者缓冲器 1025 反过来又向 DMEM 405 提供协议数据单元。RxByte 处理器 1013 内部的部件包括：

§ 控制存储器 1101，它包括被处理器 1013 所执行的微代码，并且通过提供用于控制处理器 1013 中其它部件的控制信号（虚箭头），来对目前正在被访问的微指令作出响应。控制存储器 1101 可以由 XP 313 加载。

§ 下一个地址逻辑 1105，它通过从控制存储器 1101 中选择下一个需要被执行的微指令，来对来自控制存储器 1105 的控制信号和从条件代码复用器 1121 和内容可以被访问的存储器（CAM 1117）的输入作出响应。

§ 计数器 1109 和通用寄存器 1115。这些计数器跟踪包中正被处理的比特位置。有 4 个 8－比特计数器 1109（0…3），它们可以被配置为 2 个 16－比特计数器，并且如从计数器到下一个地址逻辑 1105 的输入所显示的，这个微代码可以规定与计数器值相关的动作。这些通用寄存器 1115 是 6 个 8－比特寄存器，以用于保存在处理期间所使用的数据。

§ CAM（内容可访问的存储器）1117：一个 CAM 是在图形匹配中所使用的一个存储器。在 CAM 中的每一个信元包括一个图形，并

且当与在一个 CAM 信元中所保存的一个图形匹配的数据被提供到这个 CAM 时，这个 CAM 输出这个信元在 CAM 中的位置。这个微代码也可能规定与 CAM 输出的值相关的动作。CAM 1117 被 CPRC 401 和 XP 313 所载入。在 CAM 中有 64 个 9－比特的字，并且这个 CAM 可以通过编程的方式被划分为最多 16 个“逻辑”CAM。

§ CRC 111 是用于执行循环冗余校验的专用硬件。也可以包括其它专用硬件来对包进行解扰。

§ ALU 1119 是一个包括一个圓筒旋转器（barrel rotator）的 8－比特 ALU。

如从前面可以看出的，XP 313 通过载入 CTL 存储器 1101 和 CAM 1117 来建立 Rx 字节处理器 1013 以进行工作。CPRC 401 使用一个复位信号来停止或者启动字节处理器 1013。

一旦字节处理器 1013 被设置为运行，字节处理器 1013 就将它所接收的每一个字节提供到 CAM 1117。如果发现了表示一个传送层包的开始的一个匹配，控制逻辑 1105 就开始执行这个微代码，以处理这个传送层包。计数器被设置，并且字节处理器 1013 如 CAM 1117 所发现的进一步匹配和计数器值所指示的，来处理这个包。或者可以通过（图形，掩码，长度），或者通过（偏移，图形，掩码，长度）来规定在微代码中的匹配，其中偏移是在这个包中的偏移，掩码规定“不需要关心”这个比特，长度规定匹配中比特的数目。来自这个传送层包的协议数据被提取，并且经过路径 1019 被路由到提取空间 417 或者路由到环行总线接口 415，并且这个协议数据单元被提取并且经过路径 425 被发送到 16 字节缓冲器 1025，从这，它被 DMA 传送到 DMEM 405 中的一个线。通过（偏移，长度，寄存器地址）来规定这个微代码中的提取，其中偏移又是指 在这个包中的偏移，长度是需要被提取的字段的比特长度，并且寄存器地址是通用寄存器 1115 中、这个字段将被保存在其中的一个寄存器的地址。

Rx 比特处理器 1005 的细节

继续更详细地参考 RxSDP 421 的部件所执行的功能，Rx 比特处理器 1005 与 Rx 字节处理器 1013 类似，除了 Rx 比特处理器 1005 具有一个线性反馈移位寄存器，而不是 CRC 1111 外。这个线性反馈移位寄存器可以被配置成长度可达到 32 比特，并且具有相对这个数据流的多项式和位置。这个线性反馈移位寄存器被用于产生 hash 值或者其它校验和。Rx 比特处理器 1005 在最低的层次上，处理被 RxSDP 421 所接收的字节流。这样，Rx 比特处理器 1005 可以被编程为检测 HDLC 帧和无效的序列，被编程为删除填充的 0，被编程为在一个 OC－3 数据流中发现 STS 帧和发现并删除输入以太网帧的前导码。

接收 SONET 帧模块 1007 的细节

接收 SONET 帧模块 1007 处理 SONET 帧。帧模块 1007 可以从 Rx 比特处理器 1005，或者也可以经过旁路 1015 直接从管脚接口 204（i）接收这个帧。到 SONET 帧模块 1007 的输入包括连接到管脚接口 204（i）的物理层接口芯片所恢复的接收时钟帧同步信号和包括在这个帧中的 8－比特数据。一个 SONET 帧包括传送层开销和包括这个帧负荷的一个同步负荷封装（SPE）。包括在这个传送层开销中的是指向这个同步负荷封装的一个 STS 指针。这个同步负荷封装包括路径开销字节。

接收 SONET 帧模块 1007 对这个 SONET 帧中的字节进行解扰，进行奇偶性校验，并且经过路径 1008 将这个传送层开销写入到提取空间中。接收 SONET 帧模块 1007 进一步解释这个 STS 指针以发现这个 SPE。在发现这个 SPE 后，接收 SONET 帧模块 1007 进行奇偶性校验并且将这个 SPE 的路径开销写入到提取空间中。在这个 SPE 中的负荷被进一步传递到 RxSDP 421 的部件，如这个负荷类型所要求的。例如，如果这个负荷是 ATM 信元，它们就被传递到接收同步处理器 1009。在这个优选实施方式中，接收 SONET 帧模块 1007 不对 SPE 的负荷进行解复用。

接收同步处理器 1009 的细节

接收同步处理器 1009 专用于处理 ATM 信元。一个 ATM 信元包括 53 个字节。5 个字节是包括一个虚通道指示，一个虚信道指示，一个负荷类型指示，一个信元丢失优先级指示，一般流控制信息和一个头错误校验字节的一个头部。剩余的 48 字节是负荷。接收同步处理器 1009 通过连续地对 5- 字节序列进行头错误校验，将这个序列中的第 5 个字节作为对这个序列中前 4 个字节的头错误校验字节，来决定一个字节流是一个 ATM 信元流。如果这个头错误校验失败了，接收同步处理器 1009 就继续进行尝试。如果这个头错误校验成功了，接收同步处理器 1009 就发现了一个 ATM 信元。如果接收同步处理器 1009 在一个行中发现了一个可编程的信元数目，接收同步处理器 1009 就被同步到一个 ATM 信元流。接收同步处理器 1009 保持同步，直到连续头错误校验失败的次数达到一个可编程数目，这就表示同步处理器 1009 不再同步到一个 ATM 信元流。

当接收同步处理器 1009 与一个 ATM 信元流保持同步时，接收同步处理器 1009 解析这个 ATM 信元的头部，并且将这个头部的内容输出到提取空间。接收同步处理器 1009 通过对负荷进行解扰，并且通过将一个状态字节附加到这个负荷的末尾，以使如果对这个信元进行的头部校验失败了就可以丢弃这个负荷，来进一步处理这个负荷。一般地说，从接收同步处理器 1009 中输出的负荷被提供到大 FIFO 1011，这个大 FIFO 1011 具有足够的深度来允许在进一步进行处理以前，可以对 VPI- VCI 进行 TLE 搜寻。

接收字节处理器 1013 的细节

接收字节处理器 1013 可以被编程为执行几个功能:

§ 接收字节处理器 1013 为通过 SONET 的 PPP 处理 HDLC 换码序列;

§ 接收字节处理器 1013 进行 32- 比特 CRC 校验，包括以太网和 AAL5 的帧校验序列;

§ 当信道处理器 307（i）已经被与其它信道处理器 307 进行组合来处理 1000BASE－X 千兆以太网时，接收字节处理器 1013 进行以太网的定界符识别；使用这个再循环路径，接收字节处理器 1013 可以与其它接收字节处理器 1013 进行组合，来处理 T1 和 T3 的速率。这个再循环路径也允许处理多信道 HDLC 和加密/解密；和

§ 接收字节处理器 1013 将一个状态字写入到提取空间中，当它检测到一个帧结束标记符时。

§ 接收字节处理器 1013 从一个 ATM 信元的头部中提取这个 VPI－VCI，并且经过环型总线 311 向 TLE 301 发送包括虚通道指示符和虚信道指示符的一个消息。这个 TLE 301 通过将表示具有这个 VPI－VCI 组合的 ATM 流的输出队列的一个消息返回到信道处理器 307（i），来对这个消息作出响应。

接收字节处理器 1013 处理 9－比特字。接收字节处理器 1013 的操作已经被详细描述了。

各部件合作的示例

在下面的示例中，假定 RxSDP 421 的管脚接口 204（i）连接到其上正在使用 SONET 协议发送负荷数据的一个光纤上。在 SONET 帧中的负荷是 ATM 信元，并且其头部具有一个特定 VPI－VCI 的 ATM 信元中的负荷是一个 IP 包。RxSDP 421 已经被编程为从这个 SONET 帧中提取 ATM 信元，并且处理这个 ATM 信元。

从这个 SONET 帧中来的字节首先发送到 Rx 比特处理器 1005，Rx 比特处理器 1005 再将这些字节发送到 SONET 帧模块 1007。Rx 比特处理器 1005 也检测这个帧的开始，并且向接收 SONET 帧模块 1007 发送表示字节到达的一个信号。这个接收 SONET 帧模块 1007 对这个 SONET 帧中来的负荷数据进行解扰，进行奇偶校验，并且对负荷数据进行定位。ATM 信元的负荷被发送到接收同步处理器 1009，接收同步处理器 1009 检测这个 ATM 信元，读取 ATM 信元的头部，并且将 ATM 信元中的信息发送到提取空间 417。下面，ATM 信元的负荷被

发送到接收字节处理器1013,接收字节处理器1013将ATM信元的VPI－VCI对发送到TLE 301以进行翻译，并且将来自包括在ATM包的负荷中的任何包的报头信息读取到提取空间417中。

TxSDP 427的细节：图12

TxSDP 427执行与RxSDP 421相逆的操作：即，TxSDP 427从SDRAM 229接收一个协议数据单元，并且以其目的地和管脚接口206（i）所连接的物理接口所要求的形式，来添加输出这个协议数据单元所需要的协议数据。另外，操作是分层的。其中传送层包的协议数据被添加到媒质层包的协议数据的前面。图12显示了TxSDP 427的细节。这个协议数据单元经过路径431从DMEM 405到16字节缓冲器1229，TxByte处理器1213从16字节缓冲器1229中读取这个协议数据单元；这个协议数据经过路径429从融合空间419来，并且被发送到Tx SONET帧模块1207和Tx 比特处理器1205和被发送到TxByte处理器1213。到RxSDP 421的的再循环路径在441；复用器1206，1204和1202实现了旁路路径1215。组合路径1223允许一个给定TxSDP 427将媒质层数据添加到被与这个给定TxSDP 427组合的其它TxSDP 427所产生的传送层包流。当一个给定TxSDP 427是一个组合的一部分时，通过在TxByte令牌总线1225上的这个令牌来控制TxByte处理器1213的输出。TxSDP 427的部件与RxSDP 421中名字类似的部件类似，除了它们的功能是将协议数据添加到一个协议数据单元流，而不是提取它。在功能上的这个差异的一个结果是，TxSDP 427中没有设备用于发送环型总线消息。TxStatus 639和TxCB 637具有与Rx数据作用域643的相应部件类似的功能，其中除了方向上有差异外。TxSDP ctl 615中的寄存器进一步允许CPRC 401与TxSDP 427进行通信，并且SDP模式627中的寄存器配置这个旁路路径。

这些部件如下，它们是按照其中处理输出的顺序来排列的：

§ Tx字节处理器1213可以被编程为从DMEM 405读取一个协议

数据单元，并且实现字段插入，删除和替代。Tx 字节处理器 1213 也可以被编程为通过预先考虑这个协议数据单元中 48－字节块的这个 ATM 头部，并且可选地对这个信元内容进行加扰，来产生 ATM 信元。当没有协议数据单元需要被发送时，Tx 字节处理器 1213 产生空闲的 ATM 信元。

§ 大异步 FIFO 1211 的深度是 64 个字，并且其宽带是 9 比特，并且大异步 FIFO 1211 提供处理器 1213 执行的字段插入和删除所需要的灵活性。以核心时钟速率来对 FIFO 1211 进行写入，并且也可以或者使用核心时钟速率，或者使用串行时钟速率来读取 FIFO 1211。

§ SONET 帧模块 1207 产生其中的负荷是 TxByte 处理器 1213 的输出的 SONET 帧。

§ Tx 比特处理器 1205 是一个智能的并行－串行处理器。在程序的控制下，Tx 比特处理器 1205 实现了对它所接收的数据进行字段插入，字段删除和字段替代。这个输入数据是 8 比特宽，其输出数据是一次 1，2，或者 4 个比特，这与物理接口相关。处理器 1205 包括一个通用线性反馈移位寄存器。

§ 小 FIFO 1203：数据被以核心时钟速度写入到这个 FIFO，并且被以串行时钟速率来进行读取。这个 FIFO 是 8 个字深，并且是 9 比特宽。

§ 8b/10b 编码器 1201 对数据进行 8b/10b 编码。

处理器 1213，1207，和 1205 是可以被编程的，并且它们具有与上面所描述的 Rx 字节处理器 1013 相同的通用内部结构。

将通过与用于描述 RxSDP 421 的示例相逆的一个示例来描述这些部件之间的合作：这个输入是为一个 IP 包的一个协议数据单元；这个输出是其负荷为 ATM 信元的一个 SONET 帧，ATM 信元的负荷反过来又是 IP 包。这个 IP 包最后被保存在 SDRAM 229 中，从 SDRAM 229，这个 IP 包被 DMA 传送到 DMEM 405；形成 ATM 信元和 SONET 帧所需要的这个协议数据在融合空间 419 中。以 48－字节块为单位，

从 DMEM 405 中读取这个 IP 包；Tx 字节处理器 1213 为每 48－字节块产生一个 ATM 头部，并且这个所产生的 ATM 信元被发送到大 FIFO 1211，从这个大 FIFO 1211，这个所产生的 ATM 信元被 SONET 帧模块 1207 所读取。SONET 帧模块 1207 将 ATM 信元封装作为 ATM 帧的负荷，并且添加必要的 SONET 协议数据。然后，这个 SONET 帧被输出到 Tx 比特处理器 1205，Tx 比特处理器 1205 将这个 SONET 帧进行串行化，并且将这个 SONET 帧输出到小 FIFO 1203，从小 FIFO 1203，这个 SONET 帧被发送到编码 1201，并且从编码 1201，被发送到管脚接口 206（i）。

对 PDH 电话使用信道处理器 307

在很多年内，长途电话服务提供者已经使用数字中继链路来承载长途电话呼叫。在这些系统中，这个呼叫所连接的电话所产生的音频信号被量化为 1－字节采样，并且从许多呼叫来的被量化的采样和关于这些采样的路由信息一起被复用到中继链路上。

虽然这些采样和它们的路由信息可以被看作非常简单的包，但是，在这些系统中没有包的层次结构，并且在接收端口和发送端口之间的关系是固定的。所以，SDRAM 229 中的包不需要表搜寻，描述符，队列，或者缓冲器。相反，这个接收信道处理器 307（i）利用了它与发送信道处理器（j）共享全局地址空间 321 的这个事实，并且简单地将每一个采样写入到发送信道处理器 307（j）的 DMEM 405 中的一个队列。在发送信道处理器 307（j）中的 CPRC 401 管理这个队列。

信道处理器 307 的组合：图 13，14，26－27

如前面所提到的，信道处理器 307 的结构是以 4 个信道处理器 307 组成串 309。将信道处理器组成串的这种结构允许信道处理器被组合，以使它们可以使用比一单个信道处理器 307 所能够进行的速度更快的速度来进行发送或者接收。在一个优选实施方式中，组合被用于发送和接收 OC－12c 和千兆以太网协议。对这个 OC－12c 协议，串中 4

个信道处理器中的两个信道处理器可以被用于接收数据，而另外两个信道处理器可以被用于发送数据，或者两个串可以被使用，其中一个串用于接收数据，而另一个串被用于发送数据。对千兆以太网协议，两个串被使用，一个串中的 4 个信道处理器可以被用于根据这个协议进行接收，而另一个串中的 4 个信道处理器可以被用于根据这个协议进行发送。

在这里，组合一组信道处理器来接收一个协议被称作接收组合；相应地，组合一组信道处理器来发送一个协议被称作发送组合。在接收组合中，每一个信道处理器均接收从这个协议来的所有输入，但是仅仅处理输入中的一部分。在发送组合中，每一个信道处理器接收这个协议的部分输出，并且当需要时将其部分输出到实际上向这个发送媒质提供输出的信道处理器。串中信道处理器的串行 I/O 管脚以这样一个方式被进行连线，以使串中的所有信道处理器均接收相同的串行输入。也可能这样配置组合信道处理器，以使所有组合信道处理器可以被相同的定时器来控制。最后，被实现为在共享存储器中信号标记的令牌可以被用于协调串中信道处理器的操作。组合是通过设置属于这个串的信道处理器中的配置寄存器来完成的。

图 13 给出了允许以一个优选实施方式来进行组合的结构的细节。图中显示了有 4 个信道处理器 307（j,0…3）的一个串 309（j）。到一个串的输入被按照如下的方法进行组合：每一个信道处理器 307（j,k）具有 7 个 I/O 管脚 CPP（0…6），这样就有 28 个串 I/O 管脚 1301。串 I/O 管脚 1301（0）是信道处理器 I/O 管脚 1303（0，0），CLP 1301（2）是 CPP 1303（0，1），等等，直到 CLP 1301（27）是 CPP 1303（3，6）。这些管脚这样被互联，以使在 CLP 1301（0），CLP 1301（7），CLP 1301（14）和 CLP 1301（21）中的任何一个上的输入可以同时被所有的 CPP 1303（0，0），CPP 1303（1，0），CPP 1303（2，0），和 CPP 1303（3，0）所接收。通过信元/帧组合路径 1223 来组合输出，如图 12 所显示的。如这里所显示的，从每一个 TxSDP 425（j,i）来的输出 1441，

1223 被连接到这个串中其它 TxSDP 427 中每一个的复用器 1208，并且这样，一给定 TxSDP 425（j,i）可以接收在这个串的其它 TxSDP 427 中任何一个的大 FIFO 2122 输出的输出，并且能够处理在大 FIFO 1023 后面设备中的输出。一给定 RxSDP 421 或者 TxSDP 427 如何与其串相关是通过设置 SDP 模式 627 中的比特来决定的。

每一个信道处理器 307 进一步具有允许信道处理器 307 选择 11 个时钟输入中的一个的一个时钟复用器 1307。时钟输入中的 8 个，外部全局时钟输入 1309（0…7）是数字信道处理器 IC 203 的外部时钟；两个时钟输入，CPGLC 1311 是被一个信道处理器所恢复的全局时钟输入并且被提供到其它信道处理器，并且其中的一个输入，局部时钟 1313，作为数字信道处理器 IC 203 的本地时钟。

通过 3 组令牌环来协调一个信道处理器串的处理：TxSDP 令牌环 1225 协调从这个串的 TxSDP 427 中的 TxByte 处理器 1213 的输出。仅具有令牌的这个 TxSDP 427 可以输出到大 FIFO 1211。如图 10 中所显示的，RxSDP 421 中的哪一个设备进行提供是由 3 个令牌环决定的，环 1027 是用于 Rx 比特处理器 1005，环 1017 是用于 Rx 同步处理器 1017，和环 1028 是用于 Rx 字节处理器 1013。RxSDP 421 中具有一个令牌环的、并且被使能的一个设备仅当它具有这个令牌时才提供输出。使用哪一个令牌环是与 RxSDP 421 中的哪一个设备被使能相关的。通过 TxSDP 和 RxSDP 所执行的微代码，环中用于 RxSDP 421 和 TxSDP 427 的令牌被传送和测试。信道处理器令牌环 1315 控制组合中信道处理器对全局总线 319 的写入访问。仅目前具有环 1315 的令牌的这个信道处理器能够对全局总线 319 进行写入操作。信道处理器令牌环 1315 通过串中信道处理器所共享的串存储器 503 中的信号标记来实现。

通过如图 14 所显示的，一个串中的局部和共享存储器结构来进一步支持组合。这个结构的结果是串存储器 503。串存储器 503 包括用于串 309（j）中信道处理器（0…3）中 每一个信道处理器的信道处理器局部存储器 503。每一个信道处理器存储器 503（j,i）包括用于串行

数据处理器的信道处理器 307( j,i)存储器 1403, 总线控制存储器 1405, 用于 CPRC 401 的存储器 1407, 和部分指令存储器 403. 这个串中的 每一个信道处理器 307 能够经过串路径 439 访问这个串中其它信道处理器中每一个中的 DMEM 405。对另一个信道处理器中的 DMEM 405 进行访问有一个周期的延迟。当通信处理器 203 被初始化时，这个指令存储器可以被配置，以使这个指令存储器可以作为共享 IMEM 1409 在所有 4 个信道处理器之间进行共享，或者可以分割给所有 4 个信道处理器（IMEM 403)。当这个指令存储器被配置为共享存储器 IMEM 1409 时，这个串中 4 个信道处理器中的每一个能够以一个固定的环绕顺序，每周期一次地对共享 IMEM 1409 进行给定的访问。共享 IMEM 403 允许一个信道处理器或者其一个串的编程者可以在信道处理器所需的较大程序和独立信道处理器具有更大的灵活性之间进行折衷。

组合信道处理器的操作示例: 图 26 和 27

当组合信道处理器在处理千兆以太网时，这个接收器是有 4 个信道处理器的一个串 309（i)，这个发送器是有 4 个信道处理器的另一个串 309（j)。图 26 显示了 RxSDP 421（i，0…3）如何被配置，图 27 显示了 TxSDP 427（j，0…3）如何被配置。两个串 309（i）和串 309（j）被这样配置，以使这个发送器中一个信道处理器中的接收时钟是这两个串的主接收时钟. 这个发送串中的所有信道处理器均选择外部全局时钟 1309 中的信号作为用于千兆以太网的时钟。通过串存储器 503 中的信号标记来实现这个串的 CP 307 中 CPRC 401 之间的同步。

RxSDP 组合: 图 26

如图 26 所显示的，每一个 RxSDP 被这样配置，以使不是解码器 1001，Rx 比特处理器 1005，和 Rx 字节处理器 1013 的处理器被旁路。解码器 1001 向 CPRC 401 提供一个 3 - 比特同步丢失的输出 2603. Rx 比特处理器 1005 对输入的接收是被令牌总线 1027 所控制的，并且 Rx 字节处理器 1013 的输出是被令牌总线 1028 所控制的。这样，仅当 Rx

比特处理器具有这个令牌时，Rx 比特处理器才将它对小 FIFO 1003 的内容进行处理的结果输出到大 FIFO 1011，类似地，仅当 Rx 字节处理器 1013 具有这个令牌时，Rx 字节处理器 1013 才输出它对大 FIFO 1011 的内容进行处理的结果。

在这个优选实施方式中，每一个信道处理器接收一帧千兆以太网的数据，当它具有这个令牌时，并且当它接收到这个帧时，它将这个令牌传递到这个串中的下一个信道处理器，并且处理它刚才所接收的帧。如已经描述的，处理这个数据的一个结果是关于这个数据的一个描述符。这个信道处理器经过全局总线 319 将一个排队命令写入到队列管理引擎 305 中的其邮箱 511 中，并且队列管理引擎 305 通过将这个描述符进行排队来对这个命令作出响应。最后，令牌被实现为共享存储器中的信号标记，并且被用于管理接收串的成员对全局总线的访问，以使仅当这个接收串中的一给定信道处理器具有这个令牌时，它才能够向全局总线 319 进行写入。

TxSDP 组合：图 27

图 27 显示了在输出串中的 TxSDP 427（j，0…3）如何被建立的。如从图中可以看出的，TxSDP 427（j，0…3）仅被使能的部分是 Tx 字节处理器 1213，大 FIFO 1223，和组合路径 1223。输出流中的剩余处理是其中 Tx 比特处理器 1205 和编码器 1201 和 Tx 字节处理器 1213 被使能的 TxSDP 427（j，0）所完成的。当一给定 TxSDP 427（j，k）在令牌总线 1225 上具有令牌时，其 Tx 字节处理器 1213 经过大 FIFO 和组合路径 1223 向 TxSDP 427（j，0）输出数据，然后，TxSDP 427（j，0）在复用器 128 中选择合适的输入，并且在复用器 1208 后面的、被使能的处理器中处理这个输入。

在配置 2701 中，这个串中的每一个信道处理器 307 反过来输出一个千兆以太网帧。如已经描述的，一个信道处理器 307 通过向 QME 305 发送一个出列命令来获得关于需要被发送的这个帧的协议数据单元的一个描述符，来开始进行一个传输。CP 令牌环 1315 被用于确保，这

个发送串中的信道处理器以正确的顺序获得了这个描述符。仅当这个发送串中的一个信道处理器具有 CP 令牌环 1315 中的令牌时，这个信道处理器才可以访问全局总线 319，来向队列管理引擎 305 提供一个出列命令。一旦一个信道处理器具有关于需要被输出的数据的描述符时，它可以在其 Tx 字节处理器 1213 中对这个数据开始进行处理。仅当这个 Tx 字节处理器 1213 具有令牌环 1305 所提供的令牌时，这个数据才可以被从 Tx 字节处理器 1213 输出。从 Tx 字节处理器 1213 来的输出经过组合路径 1223 被发送到 TxSDP （j，0），在 TxSDP （j，0）中，从 Tx 字节处理器 1213 来的输出被输出。另外，这个结构允许这个串中一给定信道处理器处理需要被输出的这个帧，而这个发送串中的其它信道处理器正在输出它们的帧。

OC－12c 的串基本上按照上面所描述的来进行工作，除了有 4 个信道处理器的一个串中的两个信道处理器被配置成用于进行接收，而另外两个信道处理器被配置成用于进行发送。令牌环被使用，如上面所描述的，除了当 OC－12c 被用于发送 ATM 信元时，令牌环 11017 控制接收信道处理器中的接收同步处理器 1009.

执行处理器（XP）313 的细节：图 15

XP 313 是实现了 MIPS IV 指令集的一个通用 CPU. 它执行数字通信处理器 203 中的下述功能:

§ 复位 DCP 203 并且对 DCP 203 进行初始化;

§ 将程序载入到信道处理器 307 和构造处理器 303 中，并且设置参数以进行操作;

§ 建立和维护表搜寻引擎 301 所使用的翻译表 209，并且设置表搜寻引擎 301 中的寄存器;

§ 处理异常;

§ 运行 DCP 203 的实时操作系统；和

§ 当有主机 227 出现时，与主机 227 进行交互通信。

与主机 227 进行的交互通信包括向主机 227 提供在全局地址空间

321 中的一个可变大小的窗口，并且也可以包括处理 XP 313 从主机 227 接收或者向主机 227 发送的包。

图 15 是 XP 313 的一个框图。XP 313 具有与一个信道处理器 307 相同的很多部件。与这些信道处理器类似，XP 313 连接到环型总线 311，负荷总线 313，和全局总线 319。基本处理部件是与 CPRC 401 类似的 XP RISC 核心处理器 1501。有到环型总线 311，两条局部存储器 1507 和 1508 的一个接口 1515，到负荷总线 317 的一个接口 1511，和到全局总线 319 的一个接口 1513。两个 DMEM 1507 和 DMEM 1508 均是可以经过负荷总线接口 1511 被访问的，并且在负荷总线 315 上实际上是独立的节点。仅 DMEM 1507 是可以经过全局总线接口 1513 被访问的。在信道处理器中没有而 XP 313 中有的接口包括通用目的 I/O 接口 1517，PROM 接口 1525，和 PCI 接口 1523。指令存储器 1503 具有 3 个部件：可载入指令存储器 1503，指令只读存储器 1504，和指令存储器加载器 1506，指令存储器加载器 1506 包括由 XPRC 1501 执行来从 DRAM 229 载入 IMEM 1503 的程序。XP RISC 核心 1501 所执行的代码和在执行这个代码中所使用的数据被保存在 SDRAM 229 中，并且经过 DMA 被从 SDRAM 229 传送到 DMEM 1507，DMEM 1508，和 IMEM 1503，如 XP RISC 核心 1501 所需要的。与信道处理器 307 类似，XP 313 能够访问全局地址空间 321。XP/CP 配置寄存器 1517 是 XP 313 中全局地址空间 321 的一部分。

执行处理器 313 所执行功能的细节

执行处理器 313 通过向每一个信道处理器 307 发送一个复位信号，来对在数字通信处理器 203 中所接收的一个芯片复位信号作出响应。在这以后，XP 313 开始执行初始化代码。这个初始化代码可能在前面已经经过 PCI 接口 1523 和全局总线 317 被载入到 SDRAM 229 中，或者，它可能已经被包括在连接到 PROM 接口 1521 的一个可选外部 PROM 中。这个初始化代码载入每一个信道处理器的 IMEM 403，载入一个串的共享 IMEM 149，载入 RxSDP 421 或者 TxSDP 427 所执

行的程序，并且将配置信息载入到全局地址空间 321 内的寄存器中。

一旦已经对数字通信处理器 203 进行了初始化，XP 313 就执行用于数字通信处理器 203 的一个实时操作系统，支持网络监视协议，并且处理信道处理器所发起的异常例程。XP 313 进一步使用其对全局地址空间 321 的访问来管理信道处理器，构造处理器 303，队列管理引擎 305，和缓冲器管理引擎 315。它使用这个到环型总线 311 的接口来管理表搜寻引擎 301。一旦这个管理功能在翻译表 209 中插入和删除翻译表表目 211；另一个正在管理统计表。管理翻译表 209 和缓冲器管理引擎 315 的能力给予了 XP 313 能力来配置在 DCP 203 中输入和输出端口之间的关系。

当有一个主机 227 时，XP 313 将主机的可见性给到 DCP 203 的全局地址空间中，并且能够为它读取被 TLE 301 所管理的表。XP 313 可以进一步用作对从主机 226 接收的、或者发送到主机 226 的包的一个包收发器。例如，主机 227 可以是一个互联网协议节点，所以，可以接收或者发送互联网包。作为一个包收发器，XP 313 的工作方式基本上与一个信道处理器的工作方式相同，除了其 I/O 接口是一个 PCI 总线。

构造处理器 303：图 16－20

如图 3 中所显示的，构造处理器（FP）303 管理在一个数字通信处理器 203 和一个交换构造 222 之间的接口。交换构造 222 被用于在一些交换设备，例如通信处理器 203 之间进行通信。图 19 在 1901 显示了一些 DCP 203（1…x）如何被连接的，每一个 DCP 203 通过其 FP 303（i）连接到交换构造 222。在属于一个 DCP 203（i）的一个串行输入 204（i,j）上所接收的包可以经过 FP 303（i）和交换构造 222 被路由到另一个 DCP （k），在这里，它们在 FP 303（k）中被接收并且在一个串行输出 206（k,l）上被输出。

数据作为构造帧通过交换构造 222。一个构造帧的确切形式将随着交换构造结构的不同而不同，但是构造帧一般具有如图 18 所显示的

部分:

§ 构造头 1803，包括交换构造 222 使用来进行路由并且在交换构造 222 内进行流控制的信息;

§ 帧头 1805，包括将构造帧 1801 输入到交换构造 222 的源设备向从交换构造 222 接收帧 1801 的目的设备所提供的信息；和

§ 负荷 1807，是在从网络中源设备内所接收的、并且需要被目的设备输出到这个网络的负荷。

如下面将更详细描述的，构造处理器 303 可以被编程为处理不同类型的构造帧。在一个优选的环境中，构造处理器 303 可以被编程为处理具有一固定长度（FL 1809）的帧。这个固定的长度可以在 32 字节到 128 字节之间。

从前面的讨论中可以很清楚地看出，一个构造处理器 303 基本上具有与一个信道处理器 307 的功能相同的功能，除了它从交换构造 222 而不是串行端口接收输入，并且将输出提供到交换构造 222 而不是串行端口外。这个差异具有重要的影响。首先，交换构造 222 接收并行输入，并且提供并行输出，而不是串行输入和串行输出。输入或者输出的宽度取决于交换构造；在一个优选的实施方式中，构造处理器 303 可以被编程为每个时钟周期输出宽度是 8，16，或者 32 比特的数据。

第二，构造处理器 303 必须以比信道处理器 307 的速度高得多的速率来处理数据。这样做的一个原因是输入和输出是并行的，而不是串行的；另一个原因是交换构造 222 是与其它设备进行共享的，并且一个构造处理器 303 从交换构造 222 接收数据并且向交换构造 222 提供数据的速度影响所有这样设备的速度和吞吐率。为了实现操作的必要速度，构造处理器 303 被实现为一对有限状态机。在一个优选实施方式中的这个有限状态机与具有下述特性的构造帧 1801 一起进行工作:

§ 这个帧具有一预定长度;

§ 这个数据的前面是一固定长度的构造头 1803;

§ 这个交换构造通过构造头中的一个目的地比特掩码来实现多播

(同时将一个包路由到多于 1 个的目的地);

§ 可以通过一个简单的状态机来从这个构造头中提取拥塞信息; 和

§ 一系列相关构造帧 1801 中第一构造帧 1801 的关系是确定的。

构造处理器 303 的细节: 图 16

图 16 是构造处理器 303 的一个框图; 总的说来它与图 4 的信道处理器类似, 这一点将很快就会清楚。与一个信道处理器 307 类似, 构造处理器 303 被连接到负荷总线 317, 全局总线 319, 和环型总线 311; 这样, 它可以向 SDRAM 229 提供协议数据单元, 并且从 SDRAM 229 接收协议数据单元, 可以访问全局地址空间 321, 并且可以向表搜寻引擎 301 提供消息并且从表搜寻引擎 301 接收消息。这里有 3 个主要的差异:

§ Rx 构造处理器 1617 和 Tx 构造处理器 1621 通过 32 - 比特总线 1619 和 1623 被连接到交换构造 222;

§ 交换构造控制引擎 1601 不是一个完全可编程的 RISC 处理器, 而是两个可以进行参数设置的状态机: Rx 交换构造控制引擎 1604, 用于处理在构造处理器 303 中所接收的帧 1801; 和 Tx 交换构造控制引擎 1602, 用于处理需要被从构造处理器 303 输出的帧 1801; 和

§ 交换构造控制引擎 1601 直接连接 1625 和 1627 到队列管理引擎 305, 由此实现对队列管理引擎 305 的访问, 对队列管理引擎 305 的访问随时间的变化比经过全局总线 319 来进行访问的变化少。

构造处理器 303 的操作一般与一个信道处理器 307 的操作类似。取决于交换构造 222, 构造帧 1801 在 Rx 构造数据处理器 1617 中被以 8, 16, 或者 32 - 比特块来进行接收. Rx 构造数据处理器 1617 将头 1803 和 1805 与负荷 1807 区分开。头中的某些信息被发送到提取空间 1613, 在提取空间 1613 中, 这些信息可以被 Rx 交换构造控制引擎 1604 所使用; Rx 构造数据处理器 1617 使用其它信息来产生 TLE 301 的一个消息; 这个消息被经过环型总线接口 1611 和环型总线 311 发送到 TLE 301。协议数据单元经过复用器 1605, DMEM 1603, 和负荷总线 317

被 DMA 传送到 SDRAM 229 中的一个缓冲器 231（i）。Rx 交换构造控制引擎 1604 响应 Rx 构造数据处理器 1617 所发送的环型总线消息，使用缓冲器 231（i）的缓冲器标记 233，提取空间 1513 中的头信息，和从 TLE 301 所接收的信息来产生这个协议数据单元的一个描述符 217；使用到队列管理引擎 305 的专用连接 1625，交换构造控制引擎 1601 对这个描述符执行一个排队操作。

Tx 交换构造控制引擎 1602 执行发送处理。Tx 交换构造控制引擎 1602 从队列 215 中读取描述符，这个队列 215 是队列管理引擎 305 维持的指定交换构造 222 可以达到的目的地的描述符 217 的队列。构造处理器 303 从这个队列的头中读取描述符。对每一个描述符，构造处理器 303 使用在描述符中的信息来建立融合空间 1615，使融合空间 1615 具有产生构造帧 1801 的头 1803 和 1805 所需要的信息，其中构造帧的数据由描述符的缓冲器标记 233 所规定，并且构造处理器 303 使用描述符的缓冲器标记 233 来发起从缓冲器存储器 229 经过负荷总线 317，DMEM 1603 和复用器 1605 到 Tx 构造数据处理器 1621 的 DMA 传送，然后，Tx 构造数据处理器 1621 使用在融合空间 1615 中的信息来产生头 1803 和 1805，并且使用被 DMA 传送的协议数据单元来产生负荷。当 Tx 构造数据处理器 1621 产生构造帧 1801 时，Tx 构造数据处理器 1621 经过总线 1623 以 8，16，或者 32－比特块来将构造帧 1801 输出到交换构造 222。

Rx 构造数据处理器 1617 和 Tx 构造数据处理器 1621 的细节：图 17

图 17 是 Rx 构造数据处理器 1717 和 Tx 构造数据处理器 1621 的一个细节框图。从 Rx 构造数据处理器 1717 开始，Rx 构造数据处理器 1717 包括连接到输入数据总线 1619 的一个输入 FIFO 1708，一个构造头解释器 1707，一个头－负荷分离器 1703，一个负荷 FIFO 1705，和一个头部提取器和解释器 1701。负荷 FIFO 1705 经过总线 1616 被连接到复用器 1605，并且头部提取器和解释器 1701 被路径 1614 连接

到提取空间 1613，经过路径 1616 被连接到环型总线接口 1611。部件 1701，1703，和 1707 是用与在 RxSDP 421 和 TxSDP 427 中所使用的微序列器类型相同的可编程微序列器来实现的。

一般，Rx 构造数据处理器 1617 的操作与 RxSDP 421 的操作类似，除了没有串行－并行数据转换。从交换构造 222 所接收的一个构造帧 1801 的字节首先被提供到 FIFO 1708，FIFO 1708 允许构造处理器 303 和交换构造 222 运行的时钟速率可以不同。交换构造 222 写入到 FIFO 1708 的末尾，而交换构造头解释器 1707 从 FIFO 1708 的头部中进行读取。交换构造头解释器 1707 读取构造帧头 1803，并且将构造帧头 1803 的被选择部件输出到提取空间 1613。处理的下一阶段是头部－负荷分离器 1703，头部－负荷分离器 1703 将帧头 1808 从负荷 1807 中区分开，并且将这个负荷发送到 FIFO 1705，从 FIFO 1705，这个负荷被 DMA 传送到缓冲器存储器 229。FIFO 1705 是足够地大，以能够保存这个负荷，直到能够对 DMEM 1603 进行 DMA 访问为止。然后，帧头部 1808 被提供到头部提取器和解释器 1701，头部提取器和解释器 1701 对这个头部进行解释，并且将这个头部的信息输出到提取空间 1613 和/或者环型总线接口 1611。

Tx 构造数据处理器 1621 有 3 个可编程的部件和两个 FIFO。与 Rx 构造数据处理器 1617 相同，可编程的部件是用微序列器实现的。可编程部件包括头产生器 1709，它使用交换构造控制引擎 1601 放置在融合空间 1615 中的信息来产生帧头 1805，头和负荷融合 1711，它融合头 1805 和经过路径 1620 被从缓冲器存储器 229DMA 传送来的负荷 1807，和构造帧头产生器 1715，它在帧 1801 被输出到交换构造 222 以前产生构造帧头 1803 并且将它添加到帧 1801 上。FIFO 1717 允许构造处理器 303 和交换构造 222 运行的时钟速率可以不同，并且 FIFO 1717 提供了处理对 DMEM 1603 进行访问时的延迟所需要的灵活性。

使用 DCP 203 的交换系统的结构：图 19 和 20

构造处理器 303 允许一个 DCP 203 能够很容易地与能够发送和接

收符合上面所提出限制的帧 1801 的任何交换部件进行交互式通信。图 19 和 20 显示了这很多种可能结构中的 3 个结构。结构 1901，其中一些 DCP 共享前面已经详细讨论的一个交换构造 222；在结构 1905 中，没有独立的交换构造；作为代替，交换是通过经过它们的构造处理器 303 将两个 DCP 203 互相连接而形成的。这样一个系统可以通过将几个 DCP 203 的构造处理器连接到一个总线，并且通过提供一个令牌环或者某些其它机制来控制发送 DCP 203 对总线的访问，而被扩展。在结构 2001 中，连接到交换构造 222 的不仅是一些 DCP 203，而且有 DCP 203 不处理的、用于线路接口 2003 的非－DCP 逻辑 2002。这样一个结构 2001 可以被用于将前面已有设备集成到一个采用 DCP 203 的交换系统中。

表搜寻引擎 301 和翻译表存储器 207 的细节：图 21－24

如前面已经指出的，表搜寻引擎 301 对在环型总线 311 上从信道处理器 307，构造处理器 303，和执行处理器 313 所接收的消息作出响应，在翻译表存储器 207 中的翻译表 209 上执行表搜寻操作，并且将带这个操作结果的环型总线消息返回到从其中接收消息的设备。

图 21 是翻译表存储器 207 的一个当前优选实施方式的一个细节。使用 64－比特宽的流水线突发静态 RAM 模块来实现翻译表存储器 207。模块 207 被划分为 8 个表池 2101（0…7）。这个表池的尺寸可以是不同的，并且可以被划分为不同尺寸的表项，但是一给定表池中所有的表项的尺寸必须相同。图 21 中显示了两个这样的表项：链接表项 2111 和数据表项 2119。一个表池 2101 进一步被划分为由连续表项组成的部件表 2106。图 21 中显示了这样的 2 个部件表：表池 2101（1）中的链接表 2107 和表池 2107（7）中的数据表 2117。每一个表项均具有它所属于的部件表 2106 中的一个索引；这样，LTE 2111 具有一个链接索引 2109，并且 DTE 2119 具有一个数据索引 2114。一个部件表 2106 中的一个项是通过将其索引乘以表池中项的尺寸并且将这个乘积加到其部件表 2106 的开始位置而被定位的。

一般有两个类型的部件表 2106: 链接表和数据表。这两个类型的部件表 2106 均使用与数据表中数据相关的关键字。例如，一个翻译表 209 可以将一个 ATM 包头中的 VPI/VCI 对翻译成将接收 ATM 包的描述符的队列存储器 213 中的队列 215 的号码。这个 VPI/VCI 对是关键字，并且通过这个关键字被定位的数据表项 2119 包括这个队列的号码。一个搜寻算法决定了这个关键字在翻译表中是如何被使用的。一个链接表包括其它索引表项或者数据表项的索引；它被使用，当关键字被进行翻译来对数据表项 2119 进行定位时。如从一个链接表项被用于对其它项进行定位这样一个事实可以期望的，链接表 2111 包括控制信息 2113 和链接信息 2115. 使用控制信息 2113，将关键字进行翻译来确定跟随链接信息 2115 中的哪一个索引。控制信息 2113 和链接信息 2115 的准确特点由链接表 2107 所属于的翻译表 2109 的搜寻算法来决定。数据表项 2119 包括关键字 2120 和数据 2121。当这个被翻译的关键字与关键字 2120 匹配时，然后项 2119 中的数据 2121 包括这个关键字的翻译，例如，VPI/VCI 对的队列号码。

在一个优选实施方式中，一个翻译表 209 被一个搜寻算法号码 2125 所决定。这个搜寻算法号码标识一个数据结构 2124，这个数据结构 2124 包括规定翻译 表部件表 2106 的虚拟表号码 2127 和规定翻译表 209 将使用的搜寻算法类型的一个算法标识 2129。这个虚拟表号码通过一个号码来标识一个部件表 2106，TLE 301 将这个号码分解为指向这个部件表的一个表指针 2105。与目前信道处理器和构造处理器所使用的相比，使用虚拟表号码可以 在表存储器 207 中保存更多的部件表 2106，并且允许简单地通过改变这个虚拟表号码所代表的表指针 2105 就可以将一个部件表 2106 切换到另一个部件表 2106。例如，当信道处理器和构造处理器使用一给定的部件表时，执行处理器 313 可以建立一个新的部件表，然后，简单地通过发送带一个 writereg 命令 2415 的一个环型总线消息来用这个新的表替代给定表，这个 writereg 命令 2415 改变 TLE 301 中将虚拟表号码与表指针 2105 进行相关的寄存器中的表指针 2105。

一给定翻译表 209 可以由多达 4 个部件表 2106 组成。其中一个部件表必须是一个数据表 2117; 其它的表是链接表 2107。图 21 中所显示的翻译表 209 有两个部件表: 链接表 2107 和数据 2117。部件表是由翻译表描述符 2124 中的虚拟表号码来标识的。

通过一个 hash 算法而实现的一个翻译可以用作一个翻译表 209 如何被用于将一个关键字翻译成数据的一个示例。hash 算法是众所周知的。hash 算法所做的就是将一长比特串映射到一较短的比特串。在这个情形下，长比特串是一个关键字，并且较短的比特串是一个表项的一个索引。hash 算法可以被用于翻译表 209 仅包括数据表部件 2117 的情形。当建立这个数据表 2117 时，数据表项 2119 为其包括数据的一个关键字被进行 hash，并且这个关键字的数据表项被产生在这个 hash 算法所产生的索引（i）位置上，如果这是可能的话，并且否则，在索引 i 后面第一个可用的位置上。被 hash 到索引 i 的一个关键字在下面将被称作关键字（i)。当关键字（i）被提供给 hash 算法时，hash 算法返回数据索引 2114（i)。与这个关键字相应的 DTE 2119 或者在索引 2114（i)，其这个情形下，就结束了搜寻，或者有一个 hash 碰撞，即，多于 1 个的关键字 hash 到相同的索引（i)。在这样一个情形下，数据表 2117 可以被建立以使其关键字 hash 到相同的索引（i）的 DTE 具有在 2114（i）后面的索引，所以这个搜寻算法在索引 2114（i）开始，并且将这个关键字与后面的数据表项 2119 中的关键字 2120 进行比较，直到它发现了其中关键字 2120 与这个关键字匹配的一个数据表项，或者直到它达到数据表 2117 的末尾而没有发现一个匹配，在这个情形下，它报告没有找到一个匹配的。如果需要更高的速度，对其索引 2114 产生碰撞的 LTE 建立一个链接表，并且在碰撞发生后，这个索引被应用到这个链接表。然后，这个链接表将给出与 DTE 中这个索引相应的项的索引。

表搜寻引擎 301 对环型总线消息作出响应，对搜寻表 209 执行搜寻和维护操作。表搜寻引擎 301 使用各种搜寻算法，包括 hash 算法，二进制树算法，和 Patricia 树算法，来进行搜寻。使用表项的搜寻和

索引来实现表的维护。一般，规定表维护操作的消息来自执行处理器313。

环型总线消息：图 28

在 TLE 301 和 DCP 203 的其它部件之间的所有交互通信均是通过环型总线 311 上的消息来完成的；XP 313 使用环型总线消息来建立和维护翻译表 209，包处理器使用环型总线消息来将需要被翻译的项发送到 TLE 301，并且 TLE 301 使用环型总线消息来返回翻译的结果。图 28 显示了一个优选实施方式中的环型总线消息。消息 2801 具有两个主要的部件，数据 2817 和控制 2803。数据可以是 64 比特的任何类型数据。当一个环型总线消息被路由到 TLE 301 时，数据 2817 包括一个 TLE 命令。TLE 301 执行这个命令，并且在被发送到 TLE 命令源的一个环型总线消息的数据 2817 中返回这个结果。控制 2803 具有下述字段：

§ M 字段 2805 是被硬件设置的，并且表示这个消息比 64 比特长，所以被包括在连续时隙的一系列消息中；

§ TY 字段 2807 表示这个消息的类型；共有 4 个类型：

# 没有占据：这个环型总线时隙没有包括消息；

# 这个消息是一个指示；

# 这个消息是一个证实；

# 这个消息是一个请求；

# 这个消息是一个响应。

§ LEN 字段 2809 表示数据 2817 中这个消息的长度；

§ SEQ 字段 2811 是可以被发送者所设置的一个序列号码，以使可以确定响应消息的顺序；

§ DEST 字段 2813 表示环型总线 311 上、是这个消息的目的地的设备；和

§ SRC 字段 2815，表示是源的这个设备。

指示和证实消息类型简单地被用于确定连接到这个环型总线的设

备的环型总线接口是否在工作；如果一个设备接收来自另一个设备的一个指示消息，它就向这个设备返回一个证实消息。在这个环型总线上的一个设备希望另一个设备为它执行一个操作时，它向另一个设备发送一个请求消息；当这另一个设备已经执行了这个操作时，它使用一个结果消息将这个所产生的结果发送回这个请求设备。

这样，使用一个表搜寻，希望执行表搜寻的信道处理器发送一个请求类型的请求消息，在这个请求消息中，这个信道处理器将自己规定为源，将 TLE 301 规定为目的地。数据 2817 包括这个操作的 TLE 命令，并且 SEQ 2811 可以被设置为允许信道处理器标识响应消息的一个值。TLE 301 通过执行这个消息的 TLE 命令，并且将这个执行结果在一个响应消息中发送到信道处理器，来对这个请求消息作出响应。结果在数据 2817 中，TLE 将自己规定为源，并且将信道处理器规定为目的地，SEQ 2811 中的值与它在请求消息中的值相同。

图 24 是一个表搜寻操作命令列表，当表搜寻引擎 301 经过环型总线 311 接收这些命令时将对这些命令作出响应。每一个命令在这个表中具有一行；第一列规定这个命令的内容，第二列规定其 ID 号码，第三列规定它所返回的数据，第四列规定这个命令的影响。使用 FindR 命令 2409 来实现表搜寻；剩余的命令被用于建立和维护翻译表 209，初始化表搜寻引擎 301，并且测试 TLE 是否在进行工作。

根据命令如何定位表中的项和它们对所定位的项执行的操作，这些命令可以被进行进一步的划分。关键字命令 2423 使用关键字来对项进行定位：

§ Find 命令 2405 采用一个关键字和一个算法号码作为参数，使用这个翻译表 209 和算法号码所规定的搜寻算法来搜寻这个关键字所表示的数据表项 2119，并且返回数据项 2119 的内容，或者返回一个错误，如果没有发现这个关键字的项；

§ FindW 命令 2407 采用关键字，算法号码，需要被写入的数据，以及偏移和长度说明符作为参数；它使用这个关键字和算法来发现这个关键字的数据项 2119，并且将这个长度所规定的数据长度写入到这

个项中从这个偏移位置开始的项上;

§ FindR 命令 2409 的参数与 FindW 命令 2407 的参数相同，但是从这个关键字的项 2119 中的偏移处读取数据长度并且返回这个数据。

索引命令 2421 和 2425 使用虚拟表号码和索引来定位部件表 2106 中的项。属于组 2421 的命令从命令中所规定的项读取数据并且将数据写入到这个命令中所规定的项；属于组 2425 的命令修改命令中所规定的项中的数据:

§ Write 命令 2401 具有规定一个部件表 2106 的一个虚拟表号码，规定部件表中一个项的一个索引，需要被写入的数据，规定需要被写入数据部分的一个掩码，写入开始的偏移，需要被进行写入的数据长度；它按照命令所规定的来进行数据写入

§ Read 命令 2403 具有相同的参数，除了掩码不同外；它读取在规定表的规定项的规定位置处 的数据，并且返回这个数据;

§ XOR 命令 2411 使用命令中的数据对在规定表的规定项的规定位置处 的数据执行一个 XOR 操作或者一个 CRC 操作；在执行 CRC 操作的情形下，它返回这个 CRC。

§ Add 命令 2423 将命令中的数据加到在规定表的规定项的规定位置处的数据上。

寄存器命令 2427 读取（2417）和写入（2415）TLE 301 中的寄存器；这些寄存器是用寄存器地址来规定的。这些命令被用于使用定位翻译表，部件表 2106，和搜寻算法的代码所需要的信息来对 TLE 301 进行初始化，并且被用于简单地将上下文信息写入到 TLE 301 或者从 TLE 301 中读取上下文信息。

Echo 命令 2419 简单地将命令中的数据返回到发送者；它被用于检查环型总线 311 和所连接设备的环型总线接口是否工作正确。Nop 命令 2420 是执行时不做任何处理的命令。

TLE 301 执行下述基本循环:

1 从环型总线读取一个命令;

2 执行这个命令；和

3 经过这个环型总线返回结果。

在命令是对一个表进行操作时，执行这个命令的步骤包括步骤:

a)决定表项的索引；和

b)对表项中规定位置上的数据执行所指示的操作。

对包括关键字的命令，确定表项索引的步骤包括步骤:

i.将这个关键字翻译为一第一索引值;

ii.拾取被保存在这个索引值所规定的翻译表项中的关键字;

iii.如果这个关键字和被拾取的关键字匹配，就转到步骤 V;

iv.如果它们不匹配，根据这个搜寻算法计算一个新的索引；转到步骤 ii；和

v.对被保存在这个索引值所规定的表项中的数据执行操作.

图 22 显示了表搜寻引擎 301 的内部结构。被发送到表搜寻引擎 301 的一个环型总线消息在环型总线节点 2201 中被接收；在这个消息中的命令被命令处理器 2203 和表搜寻引擎 301 中的其它部件所处理。进行这个处理所需要的信息被保存在寄存器存储器 2205 中，并且这个算法所需要的程序被保存在控制存储器 2215 中。

图 23 显示了寄存器存储器 2205 和控制存储器 2215 的细节。为了本发明的讨论方便，寄存器存储器 2205 中有 4 类感兴趣的寄存器：算法配置寄存器 2301，表配置寄存器 2311，虚拟表配置寄存器 2341，和消息上下文寄存器 2319。算法配置寄存器 2301 将命令中所使用的算法号码与表 209 和控制存储器 215 中的 hash 代码 2323 相关。对目前被 TLE 301 所使用的每一个表 207，均有一个算法配置寄存器（ACR）2301，并且 ACR 2301（i）的索引是其算法号码 3125。图 23 中显示了一单个 ACR 2301（i）。一给定 ACR 2301 包括组成表 207 的部件表的虚拟表号码。LVT1 2325 是第一索引表的虚拟表号码；LVT2 2327 是第二索引表的虚拟表号码；LVT3 2329 是第三索引表的虚拟表号码；最后，DVT 2333 是数据表的虚拟表号码。HASHF # 2331 是在搜寻虚拟表中被使用的 hash 功能的号码。

部件表配置寄存器 2311 描述了 SRAM 207 中的部件表 2106。每一个部件表 2106 均具有一个 CTCR 2311（i），并且表的 CTCR 2311 的索引是表的物理表号码 2343。每一个 CTCR 2335 表示其表的类型 2335，表项的尺寸 2337，和表在 SRAM 207 中的开始的偏移 2339。最后，VTCR 2341 描述了目前在使用的虚拟表。每一个虚拟表号码 2127 有一个 VTCR 2341，并且一给定虚拟表号码的 VTCR 2341（i）包括目前被 VTCR 2341（i）的 VT # 2127 所规定的部件表的物理表号码 2323。为了切换一给定 VT # 2127 所表示的部件表，所需要的仅仅是改变 VTCR 2341 中与 VT # 2127 相应的 PT # 2323。

消息上下文寄存器 2319 包括与目前被表搜寻引擎 301 所处理的一个环型总线消息相关的数据。有 4 个这样的消息上下文寄存器；这样，TLE 301 可以同时处理 4 个环型总线消息；等待消息可以被保存在输入 FIFO 2202 或者被保存在环型总线 311 本身上。仅一个消息上下文寄存器，消息上下文寄存器 2319（k）被显示在图 23 中。每一个消息上下文寄存器 2319 在其中包括 3 个类型的数据：来自环型总线消息的消息信息 2321，在表搜寻引擎 301 中处理环型总线消息期间被产生和使用的处理信息 2327，和包括处理的目前结果的结果 2329。当完成了处理时，这个结果将在一个环型总线消息中被返回给被处理的这个消息的源。消息信息 2321 包括命令的类型，这个命令所发送的信息，是这个环型总线消息的源的处理器，和这个消息的一个序列号。处理信息 2327 包括算法代码 2323 中的一个程序计数器，这个搜寻将检索的最后链接表项 2111 的内容，和将被拾取的下一个链接表项 2111 在 SRAM 207 中的地址。结果 2329 包括从执行这个命令而产生的信息。在一个 FindR 命令的情形下，这个结果将包括这个命令规定的、从与这个关键字相关的数据表项 2119 中 读取的数据或者如果没有发现这个关键字的数据表项 2119 就是一个空值。

返回到图 22，部件 2203，2207，2213，2209，2211，和 2219 是通过能够访问控制存储器 2215 和寄存器存储器 2205 的处理部件来实现的。部件 2209 能够经过 SRAM 存储器控制器 2217 执行对表存储器

207 的读取操作，部件 2219 能够经过 SRAM 存储器控制器 2217 执行对表存储器 207 的读取和阅读操作。当发送到 TLE 301 的一个消息出现在环型总线 311 时，环型总线节点 2201 将这个消息放置在输入 FIFO 2202 的末尾；命令处理器 2203 读取 FIFO 2202 的头部。命令处理器 2203 将来自这个消息的信息保存在一组消息上下文寄存器 2319（i）中。然后，这组消息上下文寄存器 2319（i）被其它部件所使用，当它们执行这个消息的命令时。

部件的功能如下：

§ 初始的索引产生器 2207 从一组上下文寄存器 2319（k）中的一个关键字产生一个部件表的一个初始索引；

§ 地址产生部件 2209 从这个索引产生这个部件表项的地址，并且从这个部件表项拾取一个关键字；

§ 比较和寄存器拾取部件 2211 拾取被保存在这组上下文寄存器 2319（k）中的关键字；

§ SRAM 数据寄存器 2219 比较这个被拾取的关键字和被保存的关键字，并且根据这个比较来判断是否已经发现了被保存关键字的数据表项 2119。如果已经发现了被保存关键字的数据表项 2119，SRAM 数据寄存器 2219 拾取这个数据表项，并且将带这个数据表项的内容的一个环型总线消息发送到输出 FIFO 2221；否则，SRAM 数据寄存器 2219 拾取这个链接表项，并且将它放置在上下文寄存器 2319（k）中，并且在索引产生 2213 中继续进行处理；

§ 索引产生部件 2213 使用被保存的关键字和被拾取的链接表项来产生下一个表索引 2111（1）的索引，并且将这个索引放置在上下文寄存器 2319（k）中；然后，地址产生部件 2219 使用这个索引来产生下一个表项的地址。

用包括一个 findR 命令 2409 的一个环型总线消息作为一个示例，并且假定命令中的 alg#规定了一个 hash 算法，一旦命令处理器 2203 已经在上下文寄存器 2319（k）中为这个消息建立了消息信息 2321，初始的索引产生器 2207 使用从这个命令中来的关键字执行这个 hash

算法，以获得值 l。地址产生 2209 使用 l 来计算数据表项 2119（l）的地址，并且 SRAM 数据寄存器 2219 拾取项 2119（l）。如果其关键字字段 2120 包括关键字，就完成了这个搜寻，并且 SRAM 数据寄存器 2219 产生在数据字段 2121 中包括数据的一个环型总线响应消息。否则，索引产生 2213 就将 DIX 2114 增加 1，地址产生 2208 增加下一个 DTE 2119 的地址，如上面所描述地，这下一个 DTE 2119 的地址被拾取，并且被进行测试。以这个方法继续进行执行，直到其关键字字段 2120 与关键字匹配的一个 DTE 2119 被发现或者已经达到了数据表的末尾。

表搜寻引擎 301 的其它使用

如 XOR 2411 和增加 2413 命令的出现可以很清楚地看出，TLE 301 除了可以维护表并且在表中搜寻信息外，还可以做更多的功能。因为每一个包处理器能够以一个固定的最大延迟来对 TLE 301 进行访问，TLE 301 和翻译表存储器 207 一般可以被用于保存和处理与一个包处理器所正在处理的输入包流相关的上下文信息，这样，来克服一个包处理器仅具有数量相对较少的 DMEM 405 所带来的限制。完成地址翻译所需要的信息是这样一个上下文信息的示例。另一个是检测一个包的正确性所需要的信息，其中这个包正在被作为另一个包的负荷而承载。

包中的正确性检测是通过在包末尾的一个循环冗余校验代码（CRC）来实现的。当产生这个包时，就从这个包的内容计算出 CRC，当这个包到达其目的地时，就重新计算 CRC 并且将这个 CRC 与包中所包括的 CRC 进行比较。如果它们是相同的，这个包无错误地到达的可能性就很高；如果它们是不同的，这个包已经被破坏的可能性是同样地高。在后面的情形下，这个包被丢弃，并且一个消息被发送到这个发送者以请求重新发送这个包。一个 SDP 420 必须能够计算一个包的 CRC，以校验一个输入包的 CRC 和向一个输出包提供 CRC。当一个包在被接收或者输出时，有许多已知的算法可以计算飞行的 CRC.

从前面可以很清楚地看出，在 SDP 420 中计算 CRC 需要与一个包相关的 CRC 信息在这个包通过 SDP 420 的整个时间期间内均被保持。在其 CRC 需要被计算的包是作为传送层包中负荷的高层包，并且承载属于不同高层包的负荷的传送层包在 SDP 420 所接收的包流和/或者从 SDP 420 所发送的包流中被进行交织的情形下，这个问题就变得更复杂了。在这样一个情形下，一个独立的 CRC 计算必须对每一个高层包进行。

在 DCP IC 203 中，通过使用 TLE 来处理和保存中间结果，解决了计算 CRC 的问题。当其 CRC 正在被计算的一个高层包的每一个部分通过这个 SDP 时，CPRC 401 收集计算高层包的这个部分的中间 CRC 所需要的信息，并且向 TLE 301 发送带这个信息和规定这个信息如何被应用到这个高层包前一个中间 CRC 的一个命令的一个环型总线消息。当高层包的最后一个部分通过时，带这个信息的最后一个环型总线消息被发送，并且 TLE 301 执行这个命令来完成 CRC 计算。然后，CPRC 401 发送读取这个结果的一个读取命令 2403，并将 TLE 301 在这个环型总线消息中发送的、被返回的结果与这个包末尾的结果进行比较，来确定这个包是否已经被破坏了。

TLE 301 可以被用于保存包流上下文的另一个区域是业务统计。这些统计可以响应从包处理器来的环型总线消息在 TLE 301 中被累加，然后，可以被执行处理器 313 所读取，并且或者被执行处理器 313 或者一个主机处理器 227 使用来按照 DCP 203 所属网络的当前业务状态的需要来配置 DCP 203.

队列管理引擎 305 的细节

队列管理引擎 305 按照包处理器所规定的将描述符 217 在队列 215 中进行排队，并且又按照包处理器所规定的从队列中取出描述符。当一个包处理器独自工作时，这个包处理器典型地在多于一个的队列中对描述符进行排队，但是仅从单个队列中取出描述符。当包处理器被进行组合时，所有组合包处理器一般从一单个队列中进行读取。QME

305 进一步提供与独立的排队和从队列中进行取出的操作相关的状态信息和将队列自己与包处理器相关的信息。QME 305 既不读取它进行排队的描述符，也不决定下面一给定包处理器将从哪一个队列中进行读取。队列被完全保存在 DCP 203 中，可以被保存在那里和/或者在一个外部队列存储器 213 中，或者可以被保存在一个外部排队和规划单元中被被这个外部排队和规划单元所管理。在后面的情形下，QME 305 采用一个命令接口来将排队命令和出列命令从包处理器传递到这个外部排队和规划单元，并且将命令的结果和状态信息传递回给包处理器。在这个情形下，队列被安排的方式当然完全是取决于这个外部排队和规划单元。

这里应指出的是，一个描述符的内容完全被向 QME 305 提供描述符以进行排队的包处理器所决定，并且描述符的内容被解释的方式也完全被从队列中取出描述符的包处理器所决定。这样，QME 305 是在包处理器之间有顺序 地通过消息的一个一般系统，包处理器属于 DCP 203 是其中一个部件的交换系统，并且这个一般系统被用于在包处理器和外部排队和规划单元之间传递信息。

包处理器的 QME 接口：图 29

图 29 显示了一个信道处理器 307（i）的 QME 接口 2901，但是其它包处理器具有相同的接口，除了构造处理器 303 具有其自己到 QME 305 的专用连接。从 QME 305 中的接口部分开始，QME 305 的局部存储器当然是全局地址空间 321 的部分，并且因此可以被 CP 307（i）所访问。包括在 QME 305 的局部存储器中的是队列状态信息 2902。如后面将更详细描述地，队列状态信息 2902 允许信道处理器 307（i）决定它可以从 QME 305 中的哪一个队列中可以取出描述符并且决定这些队列的条件。信道处理器 307（i）中的 CPRC 401 这样可以访问 QSI 2901 来决定下面它将从哪一个队列中取出一个描述符。在 QME 305 中，对每一个包处理器均有一个队列邮箱 2903。为了进行排队或者从一个队列中取出一个描述符，CP 307（i）经过负荷总线 317

向 CP 307（i）的 QMB 203 发送一个队列命令 2913。对一个出列命令作出响应，QME 305 经过负荷总线 317 向 CP 307（i）返回一个从队列中进行取出的消息 2907。从队列中进行取出的消息 2907 包括被从队列中取出的描述符和关于这个描述符所表示的协议数据单元的信息，并且包括从这个队列中取出描述符的队列的条件。QME 305 进一步使用全局总线 319 中多余的周期来向独立的包处理器发送关于被这个包处理器所进行服务的队列的队列状态报告（BQSR）2915。这些报告表示这个包处理器所服务的哪一个队列已经变空了，并且哪一个队列已经不是空的了。它们作为在这个包处理器的局部地址空间中队列状态寄存器 601 中被接收的队列状态报告（RQSR）2915。最后，在 CP 307（i）中有一个队列操作状态寄存器，它包括表示 QME 305 从 CP 307（i）所接收的最后队列命令的执行状态的两个比特。其 4 个可能的状态是:

§ 成功地完成和/或者空闲

§ 没有成功完成

§ 忙，正在等待执行命令

§ 忙，命令正在被 QME 执行

假定 CP 307（i）正在接收和发送包，典型地，它将如下采用接口,:在已经获得了产生和对一个接收包的描述符进行排队所需要的信息后，CP 307（i）建立一个写控制模块 610 来向 CP 307（i）的 QMB 2903（i）发送一个排队命令，检查 QOS 2911 来确保这个邮箱不处于忙的状态，并且启动发送这个排队命令的 DMA。在这样对描述符进行排队时，CP 307（i）周期性地检查 RQSR 2915 来决定它正在从其中发送包的任何一个队列是否已经变为非空。如果队列中的一个已经变为非空，CP 307（i）以刚才对排队命令所描述的方法，来发送一个出列命令。QME 305 使用 DEQM 2907 对这个出列命令作出响应，并且 CP 307（i）可以使用被包括在 DEQM 2907 中的这个描述符来发送它所表示的包。CP 307（i）可以使用被包括在 DEQM 2907 中的其它信息来规划这个描述符所表示的包的发送，或者来更新它自己关于队列状态的

拷贝。这里，应注意的是，CP 307（i）可以完成上面所描述的任何事情，而不需要访问全局地址空间 321 中 QME 305 的部分并且因此去增加全局总线 319 的负担。当然，如果 CP 307（i）需要更多的关于它对其进行写入或者从其中进行读取的队列状态的信息，它就可以访问 QSI 2902。

队列命令 2913 的细节：图 30

在一个优选实施方式中，一个包处理器可以命令 QME 305 来执行 4 个操作：

§ 队列配置；

§ 在一个队列上对一个描述符进行排队；

§ 从一个队列中取出一个描述符；和

§ 在一些队列上对一个描述符进行排队。

这些操作的命令被经过负荷总线 317 发送到 QME 305；在负荷总线 317 上的一个处理(transaction)包括两个部分：一个地址和数据。对涉及单个队列的队列命令，这个地址被用于规定操作和这个队列，如在 3001 所显示的。计数字段（CNT）3003，处理号码（T#）3005，和池标识（PI）3007 对所有负荷总线地址是共同的；CNT 3003 规定了在这个处理中正在被读取或者写入的 16－比特量的数目；T# 3005 区分从相同源到一给定目的地的多个处理；PI 3007 规定目的地，或者 BME 315 中的一个缓冲器池或者为被 BME 315 所执行的缓冲器标记操作和为 QME 305 所执行的队列操作预留的池数目。在其 PI 3007 规定了 QME 3005 的地址中，这个地址进一步包括一个操作标识 3009，这个操作标识 3009 规定了上述操作中的一个，并且对涉及一单个队列的操作，这个地址包括队列号码 3011。

这个命令的数据部分的内容随命令而改变。对配置队列命令，数据 3013 在 3015 规定了在这个地址的 3011 所规定的队列可能包括的描述符 217 的最大数目，在 3017 规定了要从其中拾取描述符的、QME 305 中的描述符池，和 3019 表示描述符允许，描述符允许 3019 规定了可

以被允许的描述符 217 的数目，但是不是实际在队列中使用的描述符 217 的数目。这个配置队列命令允许这个包处理器读取一给定队列来动态地改变被分配到这个队列的资源数目，当条件改变时。例如，如果在被正在从这个队列进行发送的信道处理器服务的输出端口上有一个突发业务，这个信道处理器 307 可以使用这个配置队列命令来增加这个队列的最大描述符数目和/或者描述符允许，并且当突发过去时，这个信道处理器 307 可以减少这个队列的最大描述符数目和/或者描述符允许。

对单播排队命令，有两个字的数据 3021。第一个包括描述符权重 3023，描述符权重 3023 规定了正在被排队的描述符所表示的 DRAM 229 中的数据量。第二个包括将在 3011 所规定的队列中进行排队的描述符 217。对这个出列命令，也有两个字的数据 3025。第一个字包括正在被从队列中进行取出的描述符的描述符权重 3023，是仍然在队列中的描述符的总权重的队列权重 3027，和是仍然被保留在队列中描述符数目的队列长度 3029。第二个字包括已经从在 3011 所规定的队列中进行取出的描述符 217。接收这个被从队列中取出的描述符的包处理器可以使用第一个词中的信息，来决定下面它将向它正在从其中进行发送的哪一个队列发送一个出列命令，或者发送一个配置队列命令来改变一个队列可以使用的资源数目。

其地址部分被显示在 3031、其数据部分被显示在 3035 中的多播排队命令对将被多于一个包处理器发送的一个描述符进行排队。在地址部分 3031 和地址部分 3001 之间的唯一差异是它包括队列级别（QLEV）字段 3033 而不是队列号码字段 3011。队列级别字段 3033 规定了将接收这个描述符的队列的一个最小服务或者优先级程度。这个命令的数据部分 3035 在其第一个字中包括指出哪一个包处理器将输出这个队列的一个多播矢量（MCV）3037。另外包括在第一个字中的是这个描述符 217 的描述符权重 3023，描述符 217 是在第二个字中。如这个命令中的数据所指出的，这个多播排队命令规定了包处理器和服务级别而不是具体的队列，并且 QME 305 在被具有至少最低服务级

别的所规定包处理器进行服务的队列中对描述符 217 进行排队。这个描述符实际上 没有被拷贝到所有队列，如下面将详细描述的。向 QME 305 发送一个多播排队命令的、正在进行接收的包处理器也向 BME 315 发送一个命令，这个命令设置在多播排队命令的描述符中所规定的 BT 233 的一个计数器；每一次一个正在发送的包处理器接收到已经被进行多播（在描述符中被指出）排队的一个描述符并且发送由描述符的 BT 233 所规定的 PDU 时，这个正在发送的包处理器向 BME 315 发送一个命令，以将 BT 233 的计数器减 1。

队列数据结构的细节：图 31 和 32

图 31 和 32 显示了在一个优选实施方式中，在缓冲器管理引擎 305 中实现队列 215 的数据结构的细节。与 DCP 203 正在被采用的这个特定情形所需要的队列的数目和尺寸相关，这个数据结构可以都在 DCP 203 内部的存储器中，这个队列控制数据结构可以在 DCP 203 内部的存储器中，并且队列自己可以是在队列存储器 213 中，或者所有队列数据结构均在队列存储器 213 中。

这些队列 215 是描述符记录 3111 的被链接列表。所有描述符记录 3111 的尺寸均是相同的，但是这个尺寸可以在系统进行初始化后的参数所设置。这些描述符记录被保存在一些描述符池 3109（0…q）中，池的尺寸和数目是由 QME 305 可以获得的存储器数量来决定的。属于一给定队列的描述符记录 3111 必须都是来自一单个描述符池 3109(i)。每一个描述符记录 3111 包括至少下述字段：

§ 使用计数（IUC）3139，它指出描述符目前正在其上进行排队的队列的数目；和

§ 下一个指针（NPTR）3115，它指向队列中下一个缓冲器记录。

在这个描述符正在被用于将关于一个缓冲器 231 的信息从一个接收包处理器传递到一个发送包处理器的情形下，描述符记录 3111 也包括下述字段：

§ 描述符权重（DW）3137，它指出描述符的缓冲器标记 233 所表示的缓冲器的大小；和

§缓冲器标记（BT）233，SDRAM 229 中这个描述符所表示的缓冲器的缓冲器标记；和

否则，描述符 217 的内容并且由此描述符记录 3111 的内容由作为这个描述符的源的这个包处理器来决定。例如，如果这个源包处理器正在处理其中这些包的最终目的地是一些以太网节点的一个流中的包，并且这个目的地包处理器是向这些以太网节点所属的一个 LAN 输出包的一个发送包处理器，描述符 217 将包括从这个缓冲器标记 233 所规定的缓冲器内容而产生的包的以太网地址。图 31 中也显示了一个多播列表记录（MCLR）3123 的一个池 3121。如后面将详细描述地，这些记录在多播中被使用。

每一个队列 215 被队列列表 3101 中的一个队列记录 3103 所表示。队列列表 3101 在全局地址空间 321 的缓冲器管理引擎 305 的部分中，并且因此可以被包处理器所读取。这个队列的队列号码 3105 是其队列记录 3103 在列表 3101 中的索引。队列列表 3101 被划分为部分 3107，每一个部分 3107 是用于发送包的每一个包处理器的。一给定包处理器的所有队列被列表 3101 的包处理器部分 3107 中一连续队列记录组所表示。在全局地址空间 321 的 QME 305 部分中的配置信息包括队列列表 3101 的每一个包处理器部分的基地址 3108 和这个包处理器正在读取的队列数目；所以，一个包处理器可以确定哪一个队列正在被哪一个包处理器所服务，并且给定一个队列号码，就可以发现这个队列记录 3103。队列列表 3101 进一步被 QME 305 所使用，来确定一给定包处理器接收关于哪一个队列的广播队列状态报告 2915。

每一个队列记录 3103 包括下述字段：

§头指针（HDPTR）3113，它指向目前正位于记录队列的头部的描述符记录 3111；

§末尾指针（TPTR）23113，它指向目前正位于记录队列的末尾的描述符记录 3111；

§队列长度（QL）3129，它是目前在记录队列中的描述符记录 3111

的数目；

§ 总描述符权重（TDW）3131，它是记录队列中所有描述符记录的 DW 字段 3137 中值的和；

§ 被分配的描述符允许（ADA）3133，它是比那些比实际在队列中的描述符多的、并且可以被分配到队列中的描述符的数目；和

§ 队列长度限制（QLL）3135，它规定了这个队列的最大允许长度。

应注意，ADA 3133 和 QLL 3135，与将是队列描述符的源的池 3809 一起，被这个配置队列命令所设置。

图 31 中显示了一单个队列 215（0）。队列 215（0）包括描述符记录 3111（i…k），其中描述符记录 3111（i）在这个队列的头部，并且所以被 QR 3103（0）中的 HDPTR 3113 所指向，描述符记录 3111（k）在这个队列的末尾，并且所以被 QR 3103（0）的 TPTR 3117 所指向。在描述符记录 3111（k）的后面是一个被分配描述符记录 3111（q…t）的一个链接列表 3119。这些描述符记录已经被分配到队列 215（0），但是还不是这个队列 215（0）的部分，因为它们在队列 215（0）末尾的后面。ADA 字段 3133 决定了可以在链接列表 3119 中的描述符记录的最大数目。

当通过一个配置队列命令来对一个队列 215（i）进行初始化时，QME 305 建立被分配描述符记录 3111 的一个链接列表 3119，并且设置队列记录 3103（i），以使头部指针 3113 和末尾指针 3117 指向链接列表 3119 中的第一个描述符记录 3111，并且 QL 字段 3129 被设置为 0。当 QME 305 执行这个操作时，它设置 QOS 寄存器 2911 以指出其状态。

当在一个包处理器的队列邮箱 2903 中接收到一个排队命令时，QME 305 拾取这个命令中的描述符 217，将这个描述符 217 写入到属于这个命令所规定的队列的列表 3119 中的第一个描述符记录 3111，将 QL 3129 增加 1，并且更新 TPTR 3117 来使 TPTR 3117 指向描述符记录 3111 被写入到其中的描述符记录 3111。如果在链接列表 3119 中没

有描述符记录 3111，QME 305 将 ADA 3133 中所规定的数目添加到这个列表中。另外，QME 305 使用 QOS 寄存器 2911 来指出这个操作的状态。如果这个队列 215 的队列长度在描述符 217 被进行排队以前是 0，QME 305 向这个队列的包处理器发送一个广播通知 2905，以指出这个队列现在不为空。

当接收到一个出列命令时，QME 305 使用头部指针 3113 来定位在这个队列头部中的描述符记录 3111，从这个描述符记录 3111 中读取描述符 217，更新头部指针 3113 以使头部指针 3113 指向这个队列中下一个描述符记录 3111，并且如果列表 3119 中有比 ADA 字段 3113 所允许数目少的描述符记录 3111，就将前面的头部描述符记录 3111 添加到列表 3119 中。另外，QOS 寄存器 2911 指出这个操作的状态。描述符 217 在这个命令的返回数据中被返回。如果这个被从队列中取出的描述符是这个队列中的最后一个描述符，QME 305 就发送一个 BQSR 2905，以表示这个队列现在是 空的。

多播排队和从队列中进行取出：图 32 和 33

如上面在队列命令的讨论中所指出的，多播排队命令允许一个包处理器对可以被不止一个的发送包处理器所使用的一个描述符进行排队。图 32 显示了在一个优选实施方式中是如何实现多播排队的。除了在队列列表 3101 上的队列外，队列管理引擎 305 维持一单个多播列表 3201。QME 305 将在一个多播排队命令中所接收的一个描述符 215 放置在多播列表 3201 中的一个描述符记录 3111 中，并且这个描述符保持在多播列表 3201 中，直到将发送这个描述符所表示的协议数据单元的所有包处理器已经发送了这个协议数据单元。

继续多播列表 3201 的细节，多播列表 3201 被指向列表 3201 中的第一个 DR 3111（g）的 一个多播列表记录 3203 所表示。列表 3201 中、表示一个描述符 215 的任何 DR 3111 将具有与其相关的一个或者多个多播列表记录 3123，包处理器仍然必须发送这个描述符 215 的协议数据单元。与 DR 3111 相关的这个多播列表记录 3123 保存一个指针

列表 3209；这些指针包括指向多播列表 3201 中下一个 DR 3111 的一个指针和指向单播队列中 DR 3111 的一些指针。

在图 32 中，详细地显示了 DR 3111（h）的指针。DR 3111（h）中的描述符 215 被一个多播排队命令进行排队，这个多播排队命令的结果是这个描述符 215 被在单播队列 215（i）和 215（j）中进行排队。这样，在队列 215（i）中 DR 3111（a）中的 NXTPTR 3115 指向多播列表 3201 中的 DR 3111（h），与在队列 215（j）中 DR 3111（k）中的 NXTPTR 3115 的指向一样。DR 3111（h）的 NXTPTR 3115 指向 MCLR 3123（r），MCLR 3123（r）是与 DR 3111（h）相关的第一个 MCLR。MCLR 3123（r）有 3 个指针：一个指针，指针 3211，指向多播列表 3201 中的下一个 DR 3111；另一个指针，指针 3212，指向在队列 215（i）中 DR 3111（h）后的 DR；第三个指针，指针 3213，指向 MCLR 3123（s），MCLR 3123（s）有第四个指向 DR 3111（l）的指针 3214，DR 3111（h）是在队列 215（j）中 DR 3111（h）后的 DR。

图 33 中显示了 MCLR 3123 的细节。每一个 MCLR 3123 具有两个队列指针 3301，两个队列指针 3301 中的每一个规定了一个队列号码（3203）和指向这个队列号码 3203 所规定的队列中的下一个 DR 3111（h）的一个下一个指针（NPTR）3205 和一个下一个 MCLR 指针（NXTMCLR）3213，这下一个 MCLR 指针（NXTMCLR）3213 指向与这个 DR 3111 相关的下一个 MCLR。在与多播列表 3201 中的一给定 DR 3111 相关的第一个 MCLR 中，第一个 QPTR 3301 指向这个多播列表中的下一个 DR 3111。

如从前面的讨论中可以看出的，简单地通过使每一个队列 215 中的前一个 DR 3111 指向 DR 3111（h）并且包括指向与 DR 3111（h）相关的 MCLR 3123 中的每一个队列中后面的 DR 3111 的一个指针，就可以将 DR 3111（h）变为任何数目单播队列 215 中的一个 DR。这样，这个多播排队操作就是将正在被排队的描述符 217 的一个 DR 3111 添加到列表 3201，决定这个描述符 217 需要在哪一个单播队列 215 中

进行排队，如单播队列 215 所需要的添加 MCLR 3123，如图 32 所显示的在单播队列中设置在前一个 DR 3111 中的指针，设置这个单播队列 215 中的末尾指针以使这个末尾指针指向多播列表中的这个 DR 3111，并且设置在使用计数 3139 来指出描述符 217 已经被在其中进行排队的单播队列的总数目，的这样一个事情。当一个单播队列对在多播队列中 DR 3111 后面的 DR 3111 进行排队时，它设置在 MCLR 3123 中其 QPTR 3301 中的 NPTR 3205 来指向这个新添加的 DR 3111. 下面将更详细地说明 QME 305 如何决定这个描述符将在哪一个单播队列中进行排队。

对多播列表 3201 上的一个 DR 3111 进行从队列中进行取出的操作按照如下来工作：只要在使用计数比 1 大，这个从队列中进行取出的操作将按照与对不在多播列表 3201 上的一个 DR 3111 所描述的方法相同的方法来工作，除了每一个从队列中进行取出的操作将在使用计数减 1 并且在新末尾 DR 3111 中的 NPTR 3115 被从单播队列的 QPTR 3301 中的 NPTR 3205 所设置外。当 DR 3111 中的在使用计数是 1 时，这个从队列中进行取出的操作另外将设置这个 DR 3111 的在使用计数为 0，并且设置其 NPTR 3115 指向多播列表中的下一个 DR 3111，并且将其 MCLR 3123 返回给一个自由列表。

在多播排队中选择单播队列

将回忆起，这个多播排队命令不规定这个描述符将在其中进行排队的队列，而是规定了发送包处理器（MCV 字段 3037）和一个队列或者服务级别（QLEV 3033）。一给定队列或者服务级别的意义完全是由读取这个队列的这个包处理器被编程的方式所决定的。在执行多播排队命令中，QME 305 必须将这个信息翻译成一个单播队列号码。这是通过队列号码映射表（QNMT ）3303 来完成的，如图 33 所显示的。

作为一个发送包处理器如何使用这个队列或者服务级别的一个简单示例，如果这个服务级别简单地反映了队列中的优先级，并且较大的号码表示较高的优先级，然后，这个发送包处理器将不会向一给定

优先级的队列提供服务，只要在高优先级的队列中存在一个非空的队列。

队列号码映射表 3303 具有针对每一个包处理器的一个部分 3307。这些部分是根据包处理器号码的顺序来进行排序的。每一个部分 3307 具有 每一个服务级别的一个项（QNMTE）3305。在一个优选实施方式中，有 8 个服务级别。包处理器 0 的一个示例部分 3307 被显示在图 33 中，在图 33 中，假定包处理器 0 使用一个简单的优先级系统，在这个简单的优先级系统中，较高的服务级别表示较高的优先级，并且包处理器 0 具有服务级别为 0，3 和 4 的队列。其中包处理器 0 没有队列的一个服务级别的一个项 33305 的值是 0；包处理器 0 有队列的一个服务级别的一个项 33305 的值包括 QROFF 和具有这个项服务级别的包处理器队列的数目，QROFF 是这个服务级别的队列的队列记录块 3103 在队列列表 3101 中的偏移。

QNMTE 3305 被包处理器号码和排队级别号码来服务，如 3309 所显示的。这样，如果地址 3309 规定了包处理器号码 0 和排队级别 3，这个地址所定位的项 3305 就是 3305（0，3）。使用在项 3305（0，3）中所规定的 QROFF，QME 305 可以发现这个包处理器和排队级别的第一个队列记录 3103；它可以选择使用这个队列，或者它可以选择具有这个排队级别的队列的另一个队列记录 3103 所规定的一个队列。如果地址 3309 规定了具有一个 0 值的一个 QNMTE 3305，表示这个包处理器没有这个排队级别的队列；QME 305 就移动部分 3307，直到它发现一更高排队级别的一个 QNMTE 3305 并且如刚才所描述地选择了这个排队级别的一个队列。

使用 QME 305 的队列管理：图 34

如前面所提到的，QME 305 所管理的队列 215 可以被完全包括在 DCP IC 203 的 QME 305 存储器中，可以被包括在一个扩展的队列存储器 213 中，或者可以对来自 QME 305 的命令作出响应而被一个外部的排队和规划单元所管理。图 34 给出了具有和不具有外部规划单元

的结构示例。在这些图中，描述符 217 的流是用实的、粗浅灰色箭头表示；协议数据单元流是用虚的、黑灰色箭头表示。

3401 显示了一个独立的 DCP 203，其中队列被包括在直接被 QMU 305 所管理的存储器中；3403 显示了一个独立的 DCP，其中已经被添加了一个外部排队和规定单元 3405；在这样的结构中，这个外部排队和规定单元处理这样的事情，例如一给定包处理器的队列数目，这些队列的级别，和多播。具有一个外部排队和规定单元的 QME 305，QME 305 在 DCP 203 中仅有其自己的存储器；在这个存储器中，有将被发送到外部单元 3405 和每一个发送包处理器的队列的描述符 217 的一单个队列，这每一个发送包处理器接收从外部单元 3405 被发送到 QME 305 以被发送包处理器所发送的描述符 217。这些队列的功能是在这个包处理器和外部排队与规划单元之间提供一个缓冲器。

3407 显示了一个结构，其中两个 DCP 203（0 和 1）被一个交换构造 222 所连接，并且这两个 DCP 203 的队列管理是被 DCP 203（1）中的 QME 305（1）所完成的。QME 305（0）简单地发送关于需要在被构造处理器 303（0）进行读取的一个队列上进行排队的描述符 217 的排队命令，构造处理器 303（0）经过交换构造 222 向构造处理器 303（1）发送命令。然后，QME 305（1）在这个命令中所指出的队列 215 上对描述符进行排队。这个队列 215 可以是被 DCP 203（0）或者 DCP 203（1）中的一个发送包处理器所读取的一个队列。关于被 DCP 203（0）中的包处理器所读取的队列的队列状态信息被从 QME 305（1）经过构造处理器 303（1），交换构造 222，和构造处理器 303（0）发送到 QME 305（0），然后，QME 305（0）设置接收者的 QOS 寄存器 2911 或者如环境所需要的向这个接收者发送一个广播队列状态报告 2915。对出列命令，如刚才所描述地这个命令被传递到 QME 305（1），并且对这个命令作出响应而被从队列中进行取出的这个描述符，如对状态信息所描述的，被发送回 QME 305（0），并且被从 QME 305（0）发送到这个发送包处理器。

当在一个 DCP 203 中的一个包处理器所接收的一个协议数据单元

需要被从另一个 DCP 中的一个发送包处理器进行发送时，在这个发送包处理器所属的 DCP 中的 BME 315 通过经过这个构造处理器将这个缓冲器标记转发到这个进行接收的包处理器的缓冲器管理引擎，来对这个发送包处理器发送这个协议数据单元的请求作出响应，这个进行接收的包处理器的缓冲器管理引擎通过经过构造处理器向这个在进行发送的包处理器的缓冲器管理引擎提供协议数据单元，来对这个缓冲器标记作出响应，然后，这个在进行发送的包处理器的缓冲器管理引擎将这个协议数据单元提供到这个发送包处理器。

3409 显示了与 3407 类似的一个结构，除了这些队列是在被 QME 305（1）所管理的一个外部排队与规划单元 3411 中外。队列命令，状态信息，和描述符被如刚才所描述地在 DCP 203（0）和 DCP 203（1）之间进行传递，除了 QME 305（1）然后将这些命令传递到这个外部排队与规划单元 3411 并且从这个外部排队与规划单元 3411 接收状态和描述符外。3413 显示了带直接向 QME 305（0）和 QME 305（1）提供服务的一个外部排队与规划单元 3415 的一个结构。操作如上面所描述的，除了任一个 QME 305 可以为自己或者作为另一个 QME 305 的一个代理来处理外部单元 3415 外。

QME 305 的外部接口：图 35

在一个优选实施方式中，QME 305 具有一个 55-管脚的外部接口，这个外部接口可能被与一个外部同步 SRAM 存储器组或者与一个排队与规划单元一起使用，如上面所描述的。图 35 显示了这个外部接口在每一个情形下是如何被使用的。3501 显示了这个存储器外部接口。有 32 个双向数据线 3503，20 个单向地址线 3505，和 4 或者 5 个控制线 3506。使用普通的方法来写入到存储器组或者从存储器组进行读取。

3507 显示了这 55 个管脚外部接口是如何被与一个排队与规划单元 3508 一起使用的。另外，这里有 32 个双向数据线 3509，16 个双向命令线 3511，和 8 个控制线 3513。就这个接口来说，QME 305 是主控，规划器 3508 是从部件。规划器 3508 或者 QME 305 可以向另一

个发送消息，但是一个消息被发送的方向是由 QME 305 决定的。消息的传送是流控制的，即规划器 3508 和 QME 305 每一个均向另一个指出它是否可以接收消息和接收什么样的消息，并且这个发送者可能不能够发送，除非这个接收器能够接收这个消息。QME 305 也是这个接口的时钟信号源。

有 4 个可能的消息尺寸：

§ 12 字节，其中 4 个是命令，8 个是数据，在 2 个时钟周期内被发送；

§ 24 字节，其中 8 个是命令，16 个是数据，在 4 个时钟周期内被发送；

§ 36 字节，其中 12 个是命令，24 个是数据，在 6 个时钟周期内被发送；和

§ 48 字节，其中 16 个是命令，24 个是数据，在 8 个时钟周期内被发送。

这个消息的尺寸是在 DCP 203 被初始化时被决定的。当然这个消息的内容是通过 QME 305 和规划器 3508 之间的交互通信来决定的，但是对大多数部分来说，它们将包括将在被规划器 3508 所管理的队列上进行排队的、或者从被规划器 3508 所管理的队列中取出来的描述符 217。

当 QME 305 被用作所有发送包处理器的一个代理，并且这些包处理器中的每一个可能或者不可能在一给定时刻从规划器 3508 接收一个描述符时，流控制就变得有些复杂了。在一个优选实施方式中，这个包处理器可能有多达 25 个队列：一个是用于执行处理器 313，一个是用于 16 个信道处理器 317 中的每一个，8 个是用于构造处理器 303. 构造处理器 303 具有 8 个队列，因为它必须对所有经过交换构造 222 的通信进行负责，并且如上面可以看出的，这样的通信包括系统控制信息和协议数据单元。另外，需要不同类型帧的设备可能被一单个构造处理器所连接。

当与一个外部规划器 3508 一起使用时，QME 305 具有一单个接收器队列（RQ）3519，QME 305 将它从这个接收包处理器所接收的所有描述符 217 均放置在单个接收器队列（RQ）3519 中，直到这些描述符能够被输出到规划器 3508 以进行排队和每一个发送包处理器的一个发送队列（TQ ）3521。当一个发送包处理器的一个 TQ 3521 是满的时，QME 305 就不能够为这个发送包处理器的这个队列接收更多的描述符 217。

因为仅有一单个输出队列，将消息路由到规划器 3506 的流控制是简单的：当这个规划器能够接收一个消息时，这个规划器在控制 3513 中激发一个信号，并且规划器流控制寄存器 3517 表示这个信号的状态，所以 QME 305 仅需要等待发送下一个消息，直到控制寄存器 3517 这样指示。将消息路由到 QME 305 的流控制是通过规划器 3508 中的 DCP 流控制寄存器 3515 完成的，规划器 3508 中的 DCP 流控制寄存器 3515 包括 25 个流控制比特，一个比特是用于每一个可能的发送包处理器。规划器 3508 可以发送其最终目的地是一给定发送包处理器的一个消息，仅当如果在 DCP 流控制寄存器 3515 中针对这个发送包处理器的流控制比特这样指示时。QME 305 发送到规划器 3508 的每一个消息的一个部分可以被用于设置或者清除寄存器 3515 中的流控制比特，并且 QME 305 发送一个消息，当一个发送器的发送队列 3521 是满的时这个消息就设置这个发送包处理器的比特，当发送队列 3521 又有空间来保存描述符时，QME 305 就发送一个消息来清除这个比特。

虽然 QME 305 的外部接口一般被用于与队列管理设备进行通信，但是它不需要是这样的。因为一个描述符的内容完全是由产生这个描述符的包处理器来决定的，这个外部接口可以被包处理器使用来将数据写入到经过这个外部接口可以被访问的一个设备和/或者，被使用来从这样一个设备中读取数据。这个能力可以被使用的一个方面是将一个包处理器编程为一个“包取样器”，即，简单地收集关于在一个包流中包的信息的一个设备。RxSDP 421 可以被编程为从这个包流中提取每一个包所需要的信息，并且将这个信息提供到 CPRC 401，然后，

CPRC 401 将这个信息封装到一个描述符中，并且对这个描述符进行排队以使这个描述符可以被 QME 305 发送到能够保存和分析这个信息的一个外部设备中。

规划器外部接口的细节：图 36 和 37

图 36 显示了这个外部接口的独立管脚如何被与一个规划器 3508 一起使用的。在图 8 中，列标号管脚 3601 显示了组成一个组的管脚的数目，并且在这个列标号方向 3603 中的箭头显示了信息流的方向。DCP ->SCHED 表示从 QME 305 到规划器 3508 的流，并且 DCP <- SCHED 表示相反方向的流。从 8 个控制管脚 3513 开始，

§ 时钟管脚 3603，提供从 QME 305 到规划器 3508 的一个时钟信号；

§ D_flow_ctrl 3607 在每一个周期，提供从 QME 305 到规划器 3508 的 3 比特流控制信息；规划器 3508 使用在一个消息的第一个两周期内接收的 6 比特 D_flow_ctrl 3607 来设置或者清除在 D 流控制寄存器 3515 中的一个比特；

§ S_flow_ctrl 3609 是从规划器 3508 到 QME 305 的一个比特的流控制信息；QME 305 使用管脚 3609 上的值来设置 S 流控制寄存器 3517；

§ Xfer_rqst 3611 是当规划器 3508 希望向 QME 305 发送一个消息时，规划器 3508 激发的一个信号；和

§ Xfer_ctrl 3613 是 QME 305 发送给规划器 3508 的两比特，以指示在管脚上的数据和控制信号如何在下面的时钟周期中被解释；细节在下面被给出。

命令管脚 3511 是双向的；它们包括 16 个命令比特和 1 个奇偶校验比特。数据管脚 3509 也是双向的；它们包括 32 个数据比特和 1 个奇偶校验比特。

接口 3507 的操作被 Xfer_ctrl 3613 所控制。这两行的 4 个值的意

义如下：

§ 00: 不是一个其后跟着一个消息的时钟周期；

§ 01: 一个时钟周期，在两个周期后，其后面将是从 QME 305 到规划器 3508 的一个消息的第一个周期；在发送这个消息所需的最初两个周期期间，D_flow_ctrl 3607 将向规划器 3508 传送流控制信息；

§ 10: 一个时钟周期，在两个周期后，其后面将是从规划器 3508 到 QME 305 的一个消息的第一个周期；在发送这个消息的第一两个周期期间，D_flow_ctrl 3607 将向规划器 3508 传送流控制信息；

§ 11: 一个时钟周期，在两个周期后，其后面将是两个时钟周期，其中流控制信息在这两个时钟周期内经过 D_flow_ctrl 3607 被传送到规划器 3508。

如从前面的描述中可以看出的，6 比特流控制信息可能在 每一个消息内被从 QME 305 传送到规划器 3508，并且当没有消息在被传送时，可以在每 2 个时钟周期内被传送到规划器 3508。这个 6 比特 的值包括规定一个空操作的一个值和其它多个值，响应这些值，规划器 3508 针对发送包处理器的 25 个队列中的每一个队列设置或者复位 D 流控制寄存器 3515 中的独立流控制比特。

如前面所讨论的，一个消息可以是 2，4，6，或者 8 周期长，并且在每一个周期内传送 16 比特的命令数据和 32 比特的描述符数据。这个命令数据的语义取决于规划器 3508 和 QME 305 被编程的方式，除了对从规划器 3508 到 QME 305 的一个消息外，这个第一周期内的命令数据必须采取 3514 所显示的形式：前面 6 个比特必须具有图案 3615，并且最后 6 个比特必须包括这个消息需要到达的队列的号码。这个号码当然决定了这个消息将被放置在哪一个发送队列 3521 中。

图 37 显示了带 2－周期和 4－周期消息的接口 3507 的操作示例。这 2－周期消息被显示在 3701。3703 显示了这个接口的时钟周期；3613 显示了这个时钟周期内在 Xfer_Ctrl 3613 上的值；3511 显示了在命令数据管脚 3511 上的内容；3509 显示了在描述符数据管脚 3509 上的内容；3607 显示了在 D_flow_Ctrl 管脚 3607 上的内容。这样，在周期 1

中，Xfer_Ctrl 3613 被设置为 01，表示在下一个周期后的周期（周期 3）将是从 QME 305 发送到规划器 3509 的一个 2－周期消息 3702 的第一个周期。在周期 2 中，Xfer_Ctrl 3613 被设置为 00，表示在下一个周期后的周期不是一个消息的第一个周期。在周期 3 中，命令 3511 和数据 3509 包括这个消息的命令数据和描述符数据的第一个周期的内容，并且 D_flow_Ctrl 3607 包括到规划器 3508 的流控制数据的第一个周期的内容。Xfer_Ctrl 3613 再被设置为 01，表示从 QME 305 发送到规划器 3508 的另一个消息的第一个周期将从周期 5 开始。

在周期 4 中，消息 3702 的第二个半部分被发送，并且命令 3511 和数据 3509 包括命令数据和描述符数据的第二个周期的内容，并且 D_flow_Ctrl 3607 包括流控制数据的第二个周期的内容。在周期 5 和 6 中，第二个消息被发送，并且在周期 5 中，Xfer_Ctrl 3613 表示下面是一第三个消息，并且从周期 7 开始。

3705 显示了从规划器 3508 向 QME 305 发送两个 4－周期消息 3707; 在周期 1 中，Xfer_Ctrl 3613 被设置为 10，表示在周期 3 中开始的第一个消息将被发送到 QME 305; 在周期 2－4 中，Xfer_Ctrl 3613 被设置为 00，因为这个消息是 4 周期长。在周期 3－6 中，第一个消息的命令数据 3511 和描述符数据 3509 的 4 个周期的内容被发送; 仅在这个消息的前两个周期，即，周期 3 和 4 中，才发送 D_flow_Ctrl 3613。在周期 5 中，Xfer_Ctrl 3613 再被设置为 10，表示第二个消息的第一个周期将从周期 7 开始。

向规划器 3508 发送流控制信息的工作方式与发送一个 2－周期消息的工作方式相同，除了在流控制序列开始前两个周期，Xfer_Ctrl 3613 具有值 11. 在周期 3 中，这个流控制信息的前 3 个比特被在 D_flow_Ctrl 3607 上发送，并且在周期 4 中，第二个 3 比特被发送。规划器 3508 和 QME 305 忽略在命令数据 3511 和描述符数据 3509 上的值。

缓冲器管理引擎 315 和缓冲器存储器 229 的详细描述

缓冲器管理引擎 315 的基本功能是管理缓冲器存储器 229 中的缓

冲器 231，从协议数据单元被在 DCP 203 中接收的时刻到它们被从 DCP 203 发送的时刻，这些协议数据单元被保存在缓冲器存储器 229 的缓冲器 231 中。首先，下面的描述将描述缓冲器管理引擎 315 提供给包处理器的、到缓冲器存储器 229 的接口，然后描述这个接口的实现细节和 BME 315 所执行的其它功能。

BME 315 的逻辑概图：图 38

图 38 显示了到缓冲器管理引擎 315 所产生的缓冲器存储器 229 的接口。在一个优选实施方式中，缓冲器存储器 229 可以被划分为多达 32 个缓冲器池 3803。当 n 是池的数目时，这些池中的 n-1 个包括缓冲器 231，缓冲器 231 反过来又包括协议数据单元 2503。n-1 个池的数目和尺寸与缓冲器 231 的数目和尺寸是在 DCP 203 被初始化时决定的；一个池可以有高达 64K 缓冲器，并且所有这些缓冲器 231 的尺寸是相同的，在一个优选实施方式中，这个尺寸可以是从 64 字节到 64K 字节。每一个池 2803 是被一个池 ID 3819 所标识，并且这个池中的每一个缓冲器是被一个缓冲器标记 233 所标识的；在一个缓冲器 233 内部，用偏移 3802 来规定位置；这里，这个偏移规定了 PDU 3804 的开始。在一个优选实施方式中，这个偏移规定了一个 16 字节块数据的开始。

第 n 个池 2803 包括关于这些缓冲器的缓冲器标记 233。对 n-1 个缓冲器池 2803 中的每一个，有一个缓冲器标记队列 3805。一个缓冲器池 2803（i）的一个缓冲器标记队列 3805（i）包括关于缓冲器池 2803（i）中每一个缓冲器 231 的缓冲器标记项 3806，并且关于缓冲器 231（i,j）的缓冲器标记项 3806（i,j）包括缓冲器 231（i,j）的缓冲器标记 233。每一个队列 2805 具有指向这个队列头的一个指针 3807 和指向这个队列末尾的一个指针 3809。当 DCP 203 被初始化时，就建立这些队列。当 DCP 203 中的一个接收包处理器需要关于一个池 2803（i）中的缓冲器的缓冲器标记时，这个接收包处理器从这个队列 3805（i）的头部接收这些缓冲器标记；当一个发送包处理器释放缓冲器标记时，这些缓冲器标记被返回到这个队列 3805（i）的末尾。

当然，如果一个多播命令已经在比多于 1 个的队列管理引擎 315 中的队列 215 上，放置了带一给定缓冲器标记 233（i,j）的描述符，缓冲器标记 233（i,j）就不能够被返回到这个队列 3805（i）的末尾，直到缓冲器标记 233（i,j）的最后一个拷贝已经被返回。在一个优选实施方式中，通过缓冲器标记计数器 3811 来解决这个问题。对在 QME 305 中比 1 个多的队列 215 中的每一个缓冲器标记，在缓冲器标记计数器 3811 中有一个项 3813，并且这个项包括缓冲器标记目前所位于其中的队列的计数。这个项可以被使用池 ID 和缓冲器标记来访问。

当一个接收包处理器为一个描述符产生一个多播排队命令时，这个接收包处理器向 BME 315 发送一个消息，以表示这个描述符位于其中的队列的数目；发送包处理器所接收的这个描述符包括来自这个发送的 DR 3111 的 INC 值；当 INC 大于 0 时，这个包处理器向 BME 315 发送一个计数器减 1 的消息，以表示 BT 计数器 3811 中 BTAG 的计数器应被减 1；当这个计数器被减 1 到 0 时，这个缓冲器标记 233 被返回到其缓冲器标记队列 3805 的末尾。

BME 315 经过负荷总线 317 从这个包处理器接收对缓冲器 231 进行写入的命令，接收从缓冲器 231 进行读取的命令，接收获得缓冲器标记的命令，接收返回缓冲器标记的命令，和接收设置和对 BT 计数器中项进行减 1 的命令。对缓冲器进行读取和进行写入的命令的形式显示在图 39 的 3901。这些字段的意义如下：

§ CNT 字段 3903 表示在传送中，有效的、连续的 16 字节数据量的数目；

§ T# 字段 3905 被一给定包处理器用于区分总线处理；

§ 池 ID 3907 标识缓冲器池 3803（0… n-l）；

§ 偏移 3909 规定了在 BTAG 3911 所标识的缓冲器中的偏移；和

§ BTAG 3911 标识了正在被读取或者正在被写入的缓冲器 231。

池 ID 3907，偏移 3909，和 BTAG 3911 一起组成了缓冲器地址 3913。如在后面关于负荷总线的讨论中 将更详细描述的，一个命令是否是一个读取命令或者写入命令是从这个命令出现在其上的负荷总线

周期来决定的。池 ID 值 0 规定 BT 池 3803（n）和池 ID 值 0x 1F 规定关于 QME 305 的命令。对读取命令，QME 315 将规定数量的数据和处理 # 3905 从规定的缓冲器返回到进行请求的包处理器。这样，这个进行请求的包处理器可以使用这个处理号码来跟踪被返回数据对什么请求作出了响应。

一个包处理器可以对 BTAG 池 3803（n）中的 BTAG 233 执行下述 BTAG 读取操作:

§ 分配 BTAG 233; 和

§ 读取在 BT 计数器 211 中的一个 BTAG 233 的 CNT 项 2813;

这些 BTAG 写入操作是 :

§ 初始化 BTAG 233;

§ 对一个 BTAG 233 进行释放;

§ 在计数器 3811 中为一个 BTAG 233 设置一个计数器;

§ 将关于一个 BTAG 233 的一个计数器进行减 1。

这些命令的形式被显示在 3915。BT 池 ID 3907 表示其中规定了一个 BTAG 的 BTAG 池 3803（n），这个 BTAG 是在其中规定了一个计数的 BTAG 字段 3911 中，这个计数在 CNT 3903 中，并且偏移 3909 包括规定 BTAG 命令之一和一个池 ID 3919 的一个命令值 3917，这个池 ID 3919 规定了被这个 BTAG 命令所影响的 BTAG 233 所属于的缓冲器池。在这个命令需要一个响应的地方，在响应中返回这个处理号码 3905。

在 BTAG 读取命令中所采用的字段如下：在分配命令中，CNT 3903 表示正在发送命令的这个包处理器所请求的 BTAG 的数目。与这个值相关，正在进行请求的包处理器将从池 ID 3919 所规定的池中接收 8，16，24，或者 32 个 BTAG 233; 当然，BTAG 字段 3911 被忽略。BME 315 通过在负荷总线上对这个进行请求的包处理器进行的一个写入操作，来向这个进行请求的包处理器返回 BTAG 233。

在计数器读取命令中，CNT 3903 被设置为 0，BTAG 3911 包括其在 BT 计数器中的计数值将被读取的 BTAG 233，并且池 ID 3919 包括

关于 BTAG 所属于的池 3803 的池标识 3819。BME 315 通过在负荷总线上对这个进行请求的包处理器进行的一个写入操作，来返回这个计数值。

继续 BTAG 写入命令，这个初始化命令被用于设置 BTE 3806 中 BTAG 233 的值。在这个命令中，CNT 规定了正在被进行初始化的 BTE 3806 的数目；可能的数目是 8，16，24，和 32。池 ID 3919 规定了正在被进行初始化的 BTAG 233 所属于的池 3803，这样，也规定了正在被进行写入的缓冲器标记队列 3805。

这个释放命令向 BME 315 返回一单个 BTAG 233，以被再利用。在这个命令中，池 ID 3919 规定了正在被返回的 BTAG 233 所属于的缓冲器池 3803，并且 BTAG 3911 包括 BTAG 233。

在这个计数器命令中，池 ID 3919 规定了其计数器正在被设置或者被进行减 1 的 BTAG 233 的缓冲器池 ID，并且 BTAG 3911 规定了自己的 BTAG 233；在这个设置计数器的命令中，CNT 3903 包括这个计数器将被设置成的这个值。QME 315 通过在 BT 计数器中产生关于 BTAG 233 的一个 CNT 项 3813 并且将这个 CNT 项 3813 设置为这个命令中所规定的值，来对这个设置计数器命令作出响应。这个设置计数器命令被接收包处理器发送，当这个接收包处理器向 QME 305 发送带关于 BTAG 233 所表示的 PDU 的一个描述符的一个多播排队命令时。这个进行减 1 的计数器命令被发送正在被进行多播的一个协议数据单元的每一个发送包处理器所发送，当这个发送包处理器已经发送了这个 PDU 时。当正在被进行减 1 的这个计数器达到 0 时，CNT 3803 所属于的 BTAG 233 被返回给 BTAG 233 的缓冲器池的 BTQ 3805 的末尾，并且计数器 3811 中关于 BTAG 的项变为无效。

BME 315 的实现细节：图 40 和 41

除了用作对缓冲器 231 进行写入并且对缓冲器 231 进行读取和用于分配和返回缓冲器标记 233 的接口外，BME 315 还可以用作到 SDRAM 229 的一般接口。图 41 显示了 SDRAM 229 的内容。除了在

4103 的 BTAG 和缓冲器池 3803（0…n），SDRAM 229 还包括：

§ 存储器配置信息 4111，它调整 SDRAM 229 的配置；

§ 包处理器代码和数据 4109，它包括当 DCP 203 被初始化时，被 XP 313 载入到包处理器的代码和数据；对信道处理器，代码和数据包括被用于对串行数据处理器进行初始化的代码和数据。

§ 翻译表 4107 包括在对 DCP 203 进行初始化时，XP 313 载入到翻译表存储器 207 中的翻译表；

§ RTOS 4101 是被 XP 313 所执行的、关于实时操作系统的代码；XP 数据存储器 4105 包括 XP 313 在执行 RTOS 4101 中所使用的数据。

XP 313 从 RTOS 4101 和 XP 数据存储器 4105 中将指令拾取到 IMEM 1503 和 DMEM 1507 与 1508，如所需要的。

图 40 是在一个目前优选的实施方式中的 BME 315 的硬件框图。BME 315 被连接到全局总线 319 和负荷总线 317。BME 315 经过负荷总线 317 接收 BTAG 和缓冲器命令并且对它们作出响应；BME 315 在全局总线 319 上接收来自 XP 313 的存储器读取请求，并且经过负荷总线 317 对这个读取请求作出响应。包处理器的初始化是被用相同的方法完成的。

BME 315 经过这些总线而接收的每一个对一个处理的请求包括一个命令 4004 和一个地址 3913，并且写入命令还包括数据 3818。这个地址如何被解释当然与命令的类型有关。在命令分析器 4003 中对命令进行分析。对配置 SDRAM 229 的命令的处理与对其它命令的处理不同；如 4001 所显示的，配置 SDRAM 229 的命令被提供到 DRAM 配置逻辑 4035，DRAM 配置逻辑 4035 将这个数据传递到一个配置 FIFO 4037，这个数据被从配置 FIFO 4037 载入到 DRAM 配置寄存器 4030 中。

其它命令如何被处理与它们是读取命令或者写入命令或者是其它命令有关。其它命令被提供到命令 FIFO；读取命令的地址被提供到读取地址 FIFO 4013；写入命令的地址被提供到写入地址 FIFO 4021，

并且数据被提供到数据 FIFO 4017；对一个命令作出响应而被读取的数据被输出到读取数据 FIFO 4043；这些 FIFO 用于提供在 DCP 293 和 SDRAM 227 之间所需要的灵活性。在地址的情形下，地址产生模块 4011 将在缓冲器和 BTAG 命令中所使用的地址翻译成适合于 SDRAM 229 的形式；为了完成这个，地址产生模块 4011 包括规定当前如何配置 SDRAM 229 中的缓冲器的一个缓冲器配置文件。如目前所实现的，与一给定缓冲器地址 3913 相应的、SDRAM 229 中的一个地址被如下计算：

SDRAM 地址 = 池基地址（池 ID）+

（（Btag & Btag 掩码(池 ID)）>>

Btag shift（池 ID））CAT

（（偏移& 偏移掩码（池 ID）））

命令，读取地址，和读取地址分别从 FIFO 被提供到队列 4067，4015，和 4025。DRAM CTRL 4009 读取在队列 4067 的头部的命令，DRAM CTRL 4009 按照 DRAM 配置寄存器 4039 的当前设置所需要地来解释这个命令，并且向 SDRAM 229 的复用器 4025 和地址驱动器 4019 和数据收发器 4041 提供必要的控制信号。

地址产生器 4027 读取在读取地址队列 415 的头部的地址，并且地址产生器 4027 将这个地址提供到驱动器 4019 并且指出这是一个读取操作。在写入地址队列 4025 的头部的地址也被地址驱动器 4019 所读取，地址驱动器 4019 将这个地址和一个写入命令提供到地址驱动器 419。在这个相同的时刻，在写入数据队列 4029 头部中的数据被输出到数据收发器 4041，以使这个数据可以被输入到 SDRAM 229。地址产生器 4017 将高优先级提供给读取地址队列 4015，因为向一个发送包处理器提供 PDU 比将 PDU 保存在 SDRAM 229 中的时间要求更严格。

为了避免这样一个情形：一个读取操作读取在写入数据队列 4029 中正等待被覆盖写的数据并且因此获得了旧的数据，BME 315 包括一个 CAM 4023。当一个地址被写入到写入地址队列 4025 的末尾时，这

个地址的项就在 CAM 4023 中被产生；当一个地址被写入到读取地址队列 4015 的末尾时，这个地址的项也被输出到 CAM 4023；如果地址的项存在匹配，在地址产生器 4017 读取读取地址队列 4015 中的下一个地址以前，在写入地址队列 4025 中的地址的队列就被清空。

BTAG 缓存 4031 包括来自每一个 BTAG 队列 3805 的头端的 BTAG 233; 这个队列 3805 中的剩余部分在 SDRAM 229 中。当对 BTAG 233 的一个请求从一个包处理器到达时，如果可能的话，就从 BTAG 缓存 4031 来满足这个请求；否则，从 SDRAM 229 中队列 3805 的部分来满足这个请求，并且从队列 3805 中的这个部分被重新载入到这个队列的 BTAG 缓存。

BTCNT 3811 实现了 BT 计数器 3811。缓冲器标记计数命令设置，读取，和对 BTCNT 3811 中的值减 1；每一次接收到一个减 1 的 BTAG 命令，就如上面所描述的，对 BTAG 的 CNT 项的值进行减 1 。

从 SDRAM 被读取的 PDU 被输出到读取数据 FIFO 4043；来自 FIFO 4043 的输出，与来自 DRAM 配置 4035，BTAG 缓存 4031，和 BT CNT 3811 的输出一起均被提供到复用器 4046，复用器 4046 选择到读取数据队列 4045 的输出，读取数据队列 4045 又依次将这个输出输出到负荷总线 317.

环型总线 311 的细节：图 28 和 42

环型总线 311 基本上被包处理器使用来向 TLE 301 发送协议数据以进行翻译，并且从 TLE 301 接收翻译的结果。但是，环型总线 311 可以被用于向任何环型总线 311 上的节点发送消息并且从任何节点接收对这个消息的答复。在一个优选实施方式中，这些节点是包处理器和 TLE 301.

环型总线 311 被设计成向在总线节点之间的消息提供有保证的访问带宽和有上限的延迟。这个总线是 91 比特宽，其中 27 比特用于控制信息，64 比特用于从发送节点发送到接收节点的数据。这个总线被时分复用成一个数目可变的时隙，其中每一个时隙包括一个核心时钟

周期。每一个时隙被以存储段组的形式从节点传送到节点。当目前在一个节点上的时隙没有被占据时（即，没有包括环型总线消息 2801），这个节点可以将到这些节点中一个的一个消息写入到这个时隙中（其它实施方式可以允许到多个节点的消息被写入到一个时隙中）。然后，这个消息开始从节点到节点的环行，直到这个目的地节点从这个时隙中取出这个消息。

每一个节点在环型总线 311 中可以有一个到 5 个时隙来包括其源节点是这个节点的消息。如果这个节点没有使用多于 1 个的时隙，这些时隙就不存在在环型总线 311 上。从这个描述中可以看出，将一个消息从一个节点发送到总线上的另一个节点所需要的时间随这个总线上消息的数目而变化，其时间上限是当每一个节点在环型节点上具有 5 个包括消息的时隙时所需要的时间。

有 5 个类型的消息 2801。每一个消息的类型被这个时隙中类型字段 2807 的值所表示。是这个消息的源的节点被 SRC 2825 所表示，并且是这个目的地的节点是 DEST 2813 所表示。这些类型是:

§ 没有占据

§ 指示，被源节点使用来查询这个目的地节点是否对环型总线消息作出响应。这个指示不包括数据。

§ 证实，被一个指示的目的地节点使用来对这个指示的源节点作出响应。这个证实不包括数据。

§ 请求，一个主动提供的消息，其上有这个目的地节点进行操作的数据，并且在某些情形下，并且这个目的地节点向这个请求的源节点返回带操作结果的一个响应消息。

§ 响应，这个请求的目的地节点向这个请求的源节点发送的一个消息，这个消息上有这个目的地节点执行的、对这个源的一个操作的结果。

图 42 显示了环型总线接口 4201，每一个节点具有到这个环型总线的环型总线接口 4201。有两个关于其目的地是这个节点的消息的

FIFO: FIFO 4203 包括请求消息；FIFO 4209 包括其目的地是这个节点的响应消息；两个 FIFO 4203 和 FIFO 4209 被这个节点所读取。溢出流 FIFO 4211 被其源是这个节点的消息所使用，这个消息必须继续在环型总线 311 上环行，因为它们的目的地节点还没有读取它们。当其目的地是这个节点的一个消息到达了这个节点时，它被放置在其类型所要求的 FIFO 中。如果这个 FIFO 中没有空间来保存这个消息，这个消息就继续环行。

这个节点经过缓冲器 4214 将消息输出到环型总线 311，其中缓冲器 4214 从 rbus_in 4202，溢出 FIFO 4211，和请求 FIFO 4217 接收消息，缓冲器 4214 包括正在被这个节点所发送的请求消息。如果溢出 FIFO 4211 是空的，当其源节点是这个节点的一个消息被在这个节点中所接收时，它就被立即放置在缓冲器 4214 中以在它到达时所占据时隙中被输出。如果溢出 FIFO 4211 不是空的，其源节点是这个节点的新接收消息就被放置在溢出 FIFO 4211 的末尾，并且在溢出 FIFO 4211 头部中的这个消息被放置在缓冲器 4214，以在这个新接收消息到达时所占据时隙中被输出。如果新接收的消息是空的，并且溢出 FIFO 4211 不是满的，在请求 FIFO 4217 头部中的这个消息被发送到空消息的时隙中；否则，在溢出 FIFO 4211 头部中的这个消息被发送到空消息的时隙中。这个机制可以确保仅当其它节点处理了一个节点所发送的消息时，这个节点才能够经过环型总线 311 发送新消息。指示和证实是被接口 4201 在硬件级别上被处理的，并且不需要被排队。

全局和负荷总线

下面关于这些总线的实现的描述将从两个总线均使用的简单总线结构的描述开始，然后，详细描述总线本身。

请求总线和返回总线：图 43 和 44

在这个优选实施方式中，全局总线 319 和负荷总线 317 被时分复用成一单个基本总线结构。这个总线结构被显示在图 43 的 4301。总

线结构 4301 是一个被时隙化的、多信道的、共享的、仲裁总线，并且允许流水线式和重叠操作。每一个操作从占据一 5 时钟周期时隙的一个请求开始。这个总线以 166MHz 的时钟速率进行工作。这个包处理器 4303，BME 315，和 QME 305 均被连接到总线结构，并且在这里被称为总线节点。

总线结构 4301 具有两个部分，被总线节点所使用来产生总线请求并且提供关于这个请求的地址和数据的请求总线 4305，和返回总线 4317，返回总线 4317 被使用向进行请求的总线节点返回一个总线请求的结果。请求总线 4305 具有 3 个信道：两个命令/地址信道，用于承载命令和地址，即用于承载关于全局总线操作的地址和命令的全局总线命令/地址信道 4307 和用于承载关于负荷总线操作的地址和命令的负荷总线命令/地址信道 4309，和用于承载关于全局总线和负荷总线操作的数据的一个数据信道 4311。在这个优选实施方式中，每一个命令－地址信道是 32 比特宽，数据信道 4311 是 128 比特宽。返回总线 4317 具有两个信道，用于承载一个请求和这个返回数据将被返回的目的地地址的返回地址信道 4321，和承载被返回数据的返回数据信道 4319。另外，返回地址信道 4321 是 32 比特宽，并且返回数据信道 4319 是 128 比特宽。为了执行一个总线操作，能够访问一个时隙的一个节点将这个操作所需要的命令和地址放置到命令－地址信道 4307 之一上，并且将这个操作所需要的任何数据放置到请求数据信道 4311 上。当一个操作将数据返回给请求者时，总线控制 4315 将到这个返回数据源的一个请求放置在返回地址信道上，其后是返回数据将被返回的目的地地址，并且然后，返回数据的源将需要被返回的数据放置在返回数据信道 4319 上。一个节点对总线结构 4301 的访问是被总线控制 4315 所控制的。如后面将更详细描述地，总线控制 4315 向每一个节点提供一个有保证的请求总线 4305 和返回总线 4317 的带宽部分。

图 44 显示了一个总线操作是如何被一个节点所执行的。每一个时隙占据 5 个总线周期。在一个命令－地址信道 4307 或者 4309 上，在一个时隙的周期内关于这个总线的信息被显示在 4402：

§ 周期 0: 请求 4405，规定了这个操作；

§ 周期 1: 地址 4407，规定了这个操作的一个地址；

§ 周期 2: 总线授权 4409: 总线控制 4315 在这个周期返回一个信号，以表示哪一个请求节点接收到访问；

§ 周期 3: 确认 4411: 如果在前一个时隙中所规定的操作成功了，总线控制 4315 在这个周期中返回一个确认。

§ 周期 4: 地址 4413，规定了这个操作的第二个地址。

如后面将更详细描述的，地址的使用是被这个操作所定义的。

一个节点对总线结构 4301 可以执行两类一般类型的操作: 短操作，它传送 4 字节的数据，和长操作，它传送 64 字节的数据。在这些类型中的每一个中，有一个读取操作和一个写入操作。在一给定时隙 4402 中，一个包处理器 4303 可以执行一个读取操作，而另一个包处理器可以执行这个类型的一个写入操作。短操作被在全局总线命令－地址信道 4307 中所规定，而长操作在负荷总线命令－地址信道上被规定。

在图 44 中，4425 显示了短操作。在一个短操作中，在一个时隙 4402 期间这个操作被在请求命令－地址信道 4309 上进行请求，在这个时隙 4402 中的第一地址是写入地址 4415，如果在这个操作中这个数据将被写入，并且如果这个数据将被读取，最后的地址就是读取地址 4417。对这个写入操作，已经被授权对请求总线 3305 进行访问的这个节点将需要被写入到这个写入地址所规定的地址中的 4 字节数据在时隙 4402 的第 5 个时隙中被放置在请求数据信道 4311 上。

对这个短读取操作，当总线控制器 4315 已经为一个读取操作授权了总线时，它在时隙 4402 的第 4 个时隙中将关于一个节点的一个请求 4421 放置在返回地址信道 4321 上。总线控制器 4315 在下一个时隙 4402 的第 1 个时隙中将关于这个数据 4423 的返回地址放置在返回地址信道 4321 上，并且这个请求 4421 中所规定的这个节点在这下一个时隙 4402 的第 3 个时隙中将返回数据 4420 本身放置在返回地址信道 4321 上。

长操作如 4427 所显示。在时隙 4402 中、一个节点请求一个长操

作的周期期间，这个节点在周期 1 中将关于一个长读取操作的读取地址 4417 放置在请求地址总线上；这个节点在周期 4 中将关于一个长写入操作的写入地址 4415 放置在请求地址总线上。在长写入操作中，如果这个节点已经被授权进行访问，这个节点在下一个时隙 4402 的周期 1－4 中，将需要被写入的 64 字节数据 4429 以 16－字节块的形式放置到请求数据信道 4311 上。在长读取操作中，如果这个节点已经被授权进行访问，总线控制器 4315 在时隙 4402 的第 5 个周期中，将规定了这个响应节点的一个请求放置到返回地址信道 4321 上；总线控制器 4315 在下一个时隙 4402 的第 1 个周期中，将这个请求节点的地址放置到返回地址信道 4321 上；这个响应节点在这下一个时隙 4402 的第 2 个周期中，将表示在返回数据中是有效的 16－字节数据块的数目的一个计数值 4435 放置到返回地址信道 4321 上，并且这个响应节点从这下一个时隙 4402 的第 3 个周期开始，将这个返回数据 4437 以 4 个 16－字节数据块的形式放置到返回数据信道 4319 上。

在请求和返回总线上实现全局总线 319 和负荷总线 317：图 45

图 45 在 4501 显示了全局总线 319 和负荷总线 317 如何被复用到总线结构 4301。从图 4 中可以清楚地看出，时隙 4402 是 5 个周期长，而需要被写入的长数据 4429 和需要被返回的数据 4437 每一个均是 4 个周期长，并且需要被写入的短数据 4419 和需要被返回的短数据 4420 每一个是一单个周期长。

这样，就可能在总线结构 4301 上重叠短操作和长操作，如图 45 所显示的，并且由此可以将请求数据信道 4311 的每 5 个周期中的 4 个周期用于负荷数据，而第 5 个周期用于全局总线并且相同的第 5 个周期用于返回数据信道 4319。 在图 45 中，实现全局总线 319 的短操作被显示在图中的上半部分；实现全局总线 317 的长操作被显示在图中的下半部分。如图中间部分的请求命令－地址信道 4307 和 4309 的表示所显示的，负荷时隙 4507 比全局时隙 4503 提前一个时隙周期；这样，需要被写入到负荷时隙 4507 中的长数据 4429 在需要被写入到全

局总线时隙 4503 的短数据 4519 和需要被写入到全局总线时隙 4505 的短数据 4419 之间，出现在请求数据信道 4311 上。类似地，在返回总线 4317 上，需要被返回的、关于负荷请求 4509 的长返回数据 4437 在需要被返回的、关于全局请求 4503 的短数据和需要被返回的、关于全局请求 4505 的短数据之间，出现在返回数据信道 4319 上。

全局总线 317 上的总线访问和寻址：图 45

图 45 显示了关于全局总线 319 和负荷总线 317 的时隙 4402 如何被划分为偶时隙和奇时隙。一个偶时隙－奇时隙对组成一个 10－周期的时间段 4502。这个偶和奇时隙与包处理器 4303 的偶和奇组相应。组中的成员关系如下所显示：

| 包处理器 4303 | 组 |
|---|---|
| 信道处理器 307（15，13，11，9，7，5，3，1） | 奇 |
| 信道处理器 307（14，12，10，8，6，4，2，0） | 偶 |
| 执行处理器 313 | 奇和偶 |
| 构造处理器 303 TX | 奇和偶 |
| 构造处理器 303 RX | 奇和偶 |

在每一个时间段 4502 中，4 个全局总线处理被执行：

§ 在偶时隙中：一个短读取操作和一个短写入操作；

§ 在奇时隙中：一个短读取操作和一个短写入操作。

对这些类型的处理中的每一个均有一个独立的令牌。在一个组中的包处理器中，以循环方式来旋转令牌，在组中，具有关于一个操作的令牌的这个包处理器对这个操作具有最高优先级。如果这个包处理器不请求它具有其令牌的这个类型的处理，按升序排列最靠近这个令牌的请求包处理器被授权访问这个总线。一个包处理器接收对这个总线的访问的最大延迟是 100 个周期。没有包处理器产生一个写入请求的时隙被队列管理引擎 305 使用来向这些包处理器广播队列状态报告 2915。

在全局总线处理中，读取地址 4417 和写入地址 4415 是全（flat）

32－比特的地址。在返回数据信道 4319 上的地址 4423 是一个有效的比特，其后是将这个接收者标识为包处理器，BME 315，或者 QME 305 之一的一个处理器标识符。

负荷总线 317 上的总线访问和寻址：图 45

负荷总线 317 上的总线访问与上面关于全局总线 319 的描述完全相同；另外，每一个时间段 4502 被划分为一个偶时隙和一个奇时隙，并且就全局总线来说，这些包处理器被分配到奇时隙和偶时隙。另外，在一单个时间段 4502 内，有关于 4 个负荷总线处理的时隙：

§ 在偶时隙中：一个长读取操作和一个长写入操作；

§ 在奇时隙中：一个长读取操作和一个长写入操作。

如对全局总线所描述的，令牌被采用来决定在这些包处理器之间的优先级，除了 QME 305 或者执行处理器 313 没有特殊的结构外。对地址来说，长读取操作和写入操作的地址是图 39 中所显示的负荷缓冲器命令。在返回地址信道 4321 上的、关于返回负荷数据的地址与返回全局数据的地址类似，除了它还包括一个 3－比特处理号码，在数据被返回的负荷缓冲器命令中，总线控制器 4315 从处理号码 3905 拷贝这 3－比特处理号码.

DCP 203 作为一个一般的数据流处理器

虽然前面的讨论已经描述了 DCP 203 是如何被用于一个包交换的，但是该相关领域内的技术人员将很清楚，DCP 203 可以被用于处理数据流的任何应用中。通过组合，DCP 203 的信道处理器 307 可以被构造成处理串行比特流中的数据，4 比特流中的数据，和字节流中的数据，并且构造处理器 303 可以处理由 32－比特字组成的流中的数据。TLE 301 提供了用于保存和处理每一个数据流上下文信息的一个机制，并且 QME 305 提供了用于将关于流中所包括的负荷的信息从接收包括这个负荷的流的包处理器传递到发送包这个负荷的流的包处理

器的一个机制，并且提供了连接到 QME 305 的外部单元。构造处理器 303 允许 DCP 203 被连接到另一个 DCP 203，连接到一个并行总线，或者连接到一个交换构造，并且这样，允许从一些 DCP 203 构造用于处理数据流的大设备，并且允许 DCP 203 与其它处理数据流的设备组合在一起。

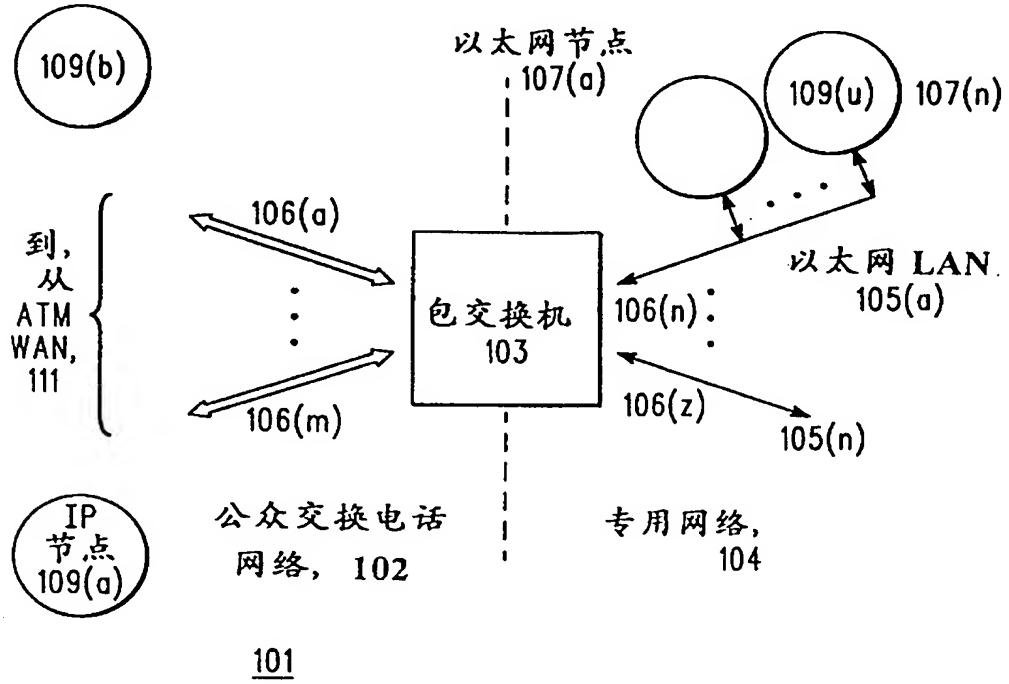
包处理器可以被编程为处理任何类型的数据流。一个可编程 SDP 420 与每一个包处理器中的一个可编程 CPRC 401 和一个 DMA 引擎的组合允许提取流中控制数据的操作与提取流的负荷，处理这个控制数据和将这个负荷传送到 BME 315 的操作分开。在 CPRC 401 内使用数据作用域来维护关于 SDP 420 和 DMA 引擎对流的处理的当前状态的信息允许对控制数据的处理可以与负荷在 BME 315 和 SDP 420 之间的传送一起同时进行，并且也大大地简化了 CPRC 401 的编程。SDP 420 中的发送处理器和接收处理器可以被编程为对输入流中的图案和对比特计数作出响应，并且这个旁路的存在允许能够比较容易地配置发送处理器和接收处理器来处理不同类型的流。通过配置一个 SDP 重新循环一个流，组合信道处理器来处理高速串行流或者由 4 比特或者字节组成的流，和配置 I/O 管脚与不同类型的发送媒质一起工作，就获得了更多的灵活性。

DCP 203 通过使用能够确保在包处理器和 TLE 301 之间通信的最小延迟的一个环型总线，通过使用一个时隙化的总线来传送突发数据，以在 BME 315 和包处理器之间传送负荷，以从 BME 315 传送缓冲器标记和将缓冲器标记传送到包处理器，和以在包处理器和 QME 305 之间传送描述符，DCP 203 处理在数据流处理中所固有的时间限制。通过允许这些设备对相互之间的局部存储器进行访问的一个全局地址，就可以在包处理器，QME 305，和 BME 315 之间进行协调。在包处理器串的情形下，串的成员能够快速地访问相互之间的局部存储器。

结尾

前面的详细描述已经向在本发明所属于的领域内的那些技术人员，公开了对本发明者目前所知为最佳的、采用它们的技术来处理数据流的模式，数据流是被发送到被设计成处理和路由包的一个数字通信处理器集成电路。相关领域内的技术人员能够立即理解，数字通信处理器的独立特征可以根据实际需要采用，而不是在这里所公开的情形中所采用，并且可以使用不同的方法从这里所公开的情形进行组合。相关领域内的技术人员将进一步认识到，可以使用不同的实现方法来实现这些特征。因为所有前面的原因，这个详细描述在所有方面均被认为是示例性的，并且不具有任何限制性，并且这里所公开的本发明的宽度不是从这个详细描述来决定的，而是从被理解为专利法所允许的全限制范围的权利要求书来决定的。

# 说 明 书 附 图



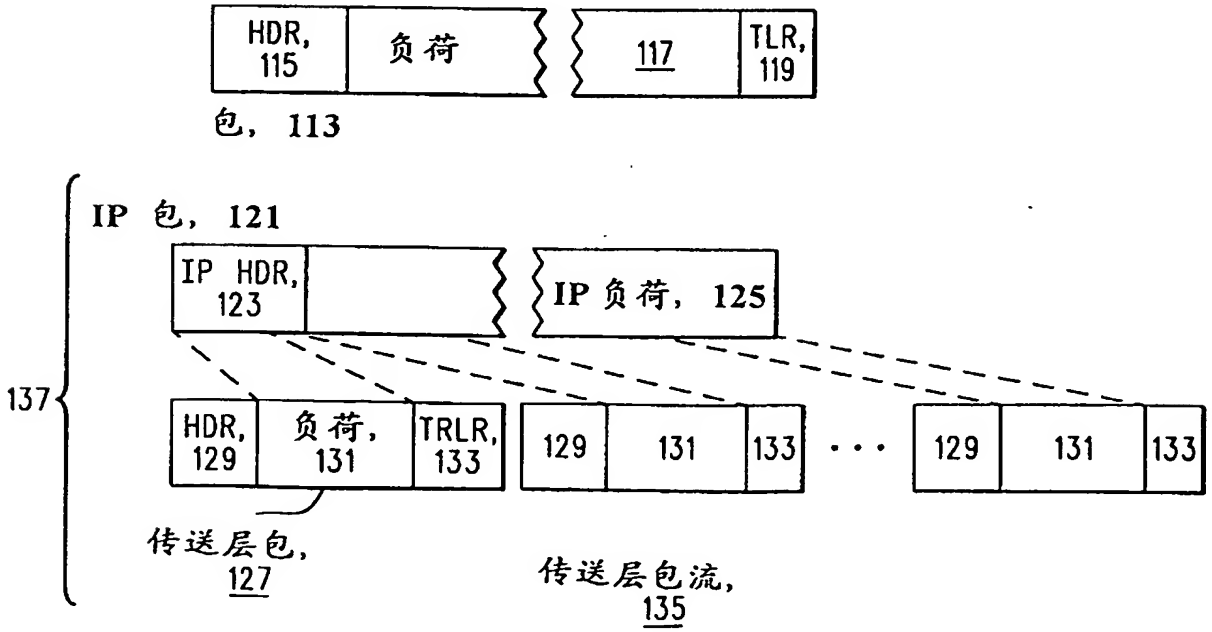




图 1
（现有技术）

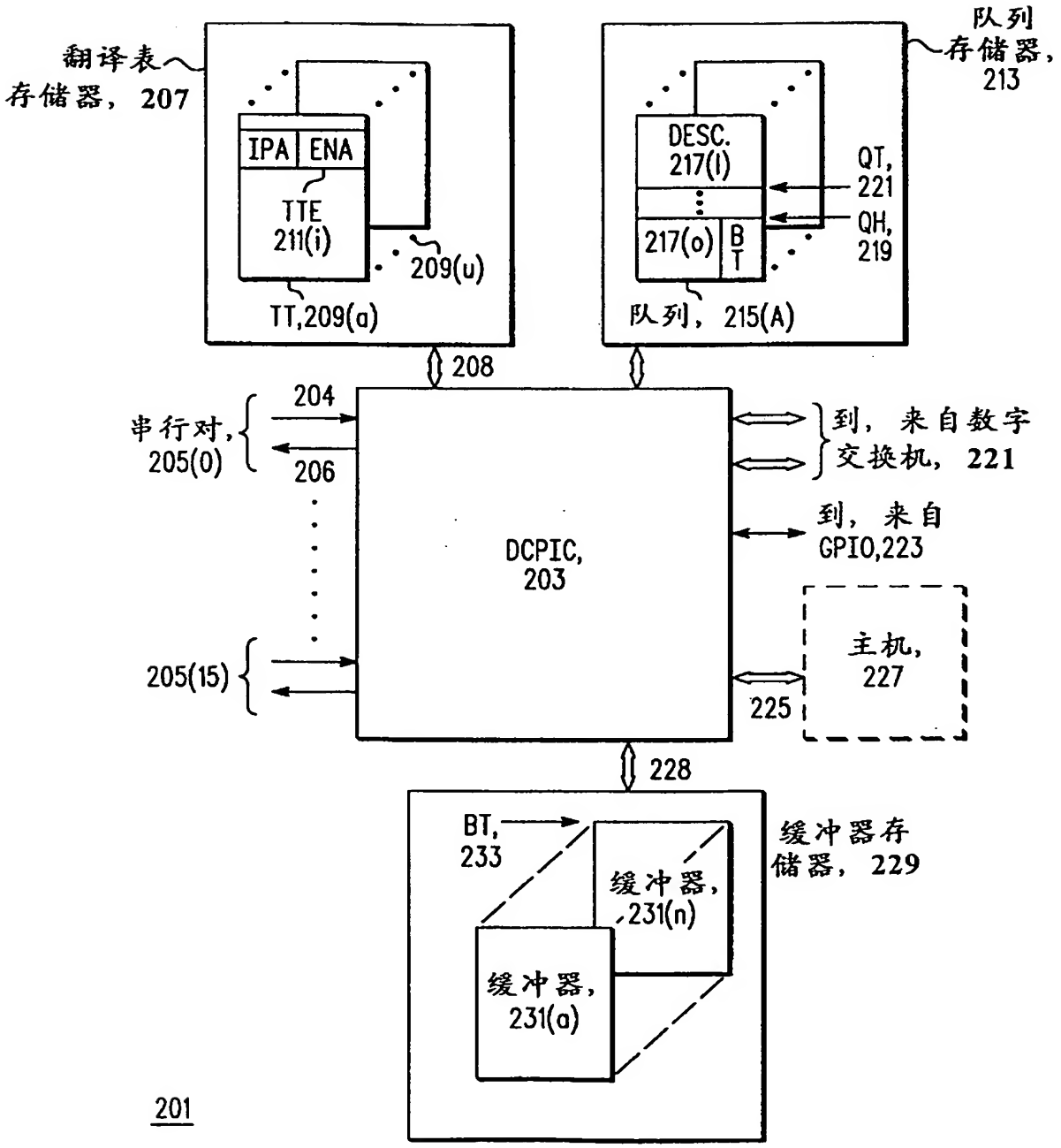

翻译表
存储器，207
IPA
ENA
TTE
211(i)
209(u)
TT,209(a)
队列
存储器，
213
DESC.
217(l)
QT,
221
QH,
219
217(o)
B
T
队列，215(A)
208
204
串行对，
205(0)
206
DCPIC,
203
到，来自数字
交换机，221
到，来自
GPIO,223
主机，
227
205(15)
225
228
BT,
233
缓冲器，
231(n)
缓冲器，
231(a)
缓冲器存
储器，229
201

图 2

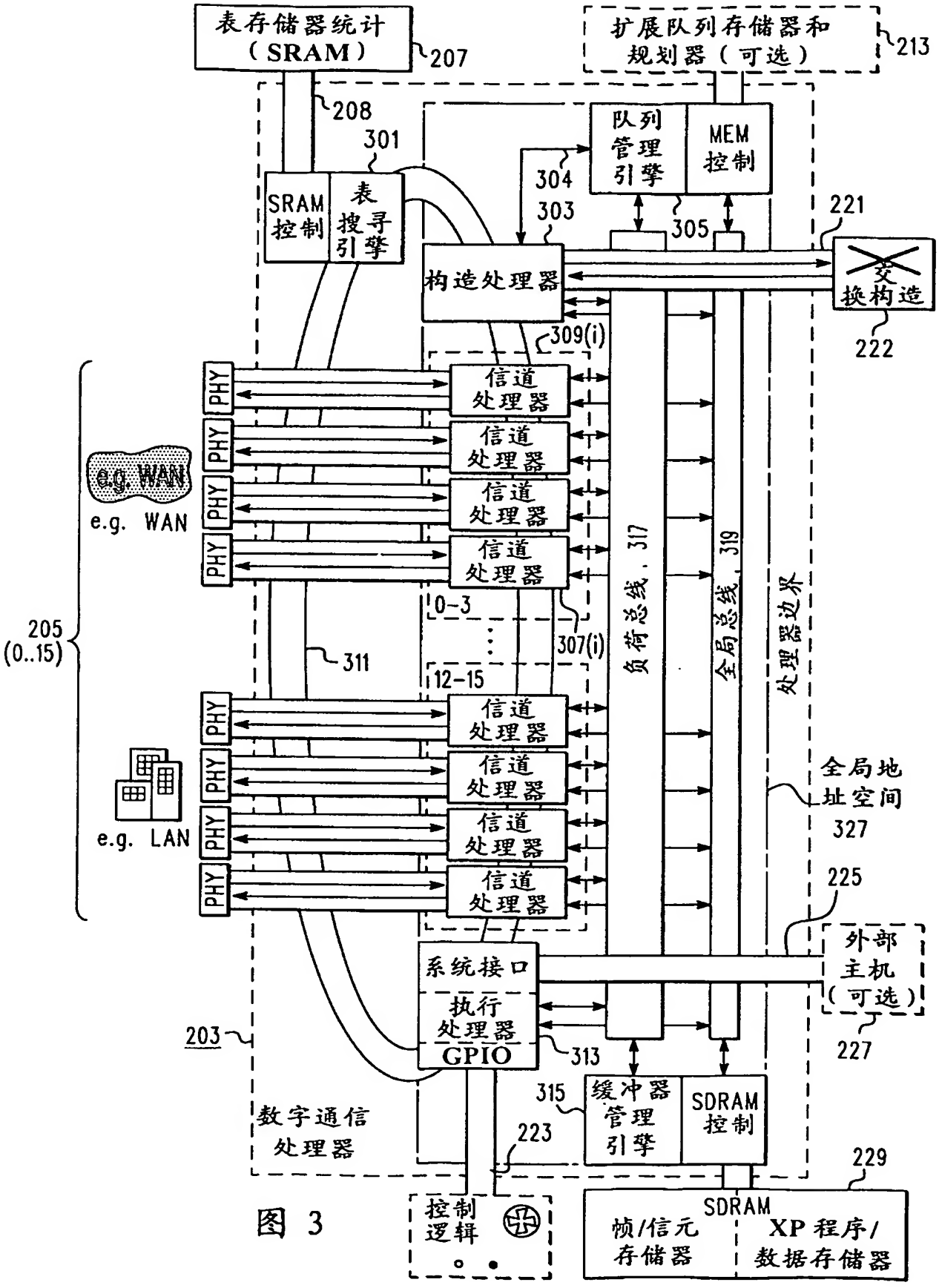

表存储器统计
（SRAM）
207
扩展队列存储器和
规划器（可选）
213
208
301
SRAM
控制
表
搜寻
引擎
304
303
队列
管理
引擎
MEM
控制
305
221
交
换构造
222
构造处理器
309(i)
信道
处理器
PHY
e.g. WAN
205
(0..15)
0-3
307(i)
311
12-15
负荷总线，317
全局总线，319
处理器边界
全局地
址空间
327
e.g. LAN
225
外部
主机
（可选）
227
系统接口
执行
处理器
GPIO
313
203
315
缓冲器
管理
引擎
SDRAM
控制
数字通信
处理器
223
229
控制
逻辑
SDRAM
帧/信元
存储器
XP 程序/
数据存储器

图 3

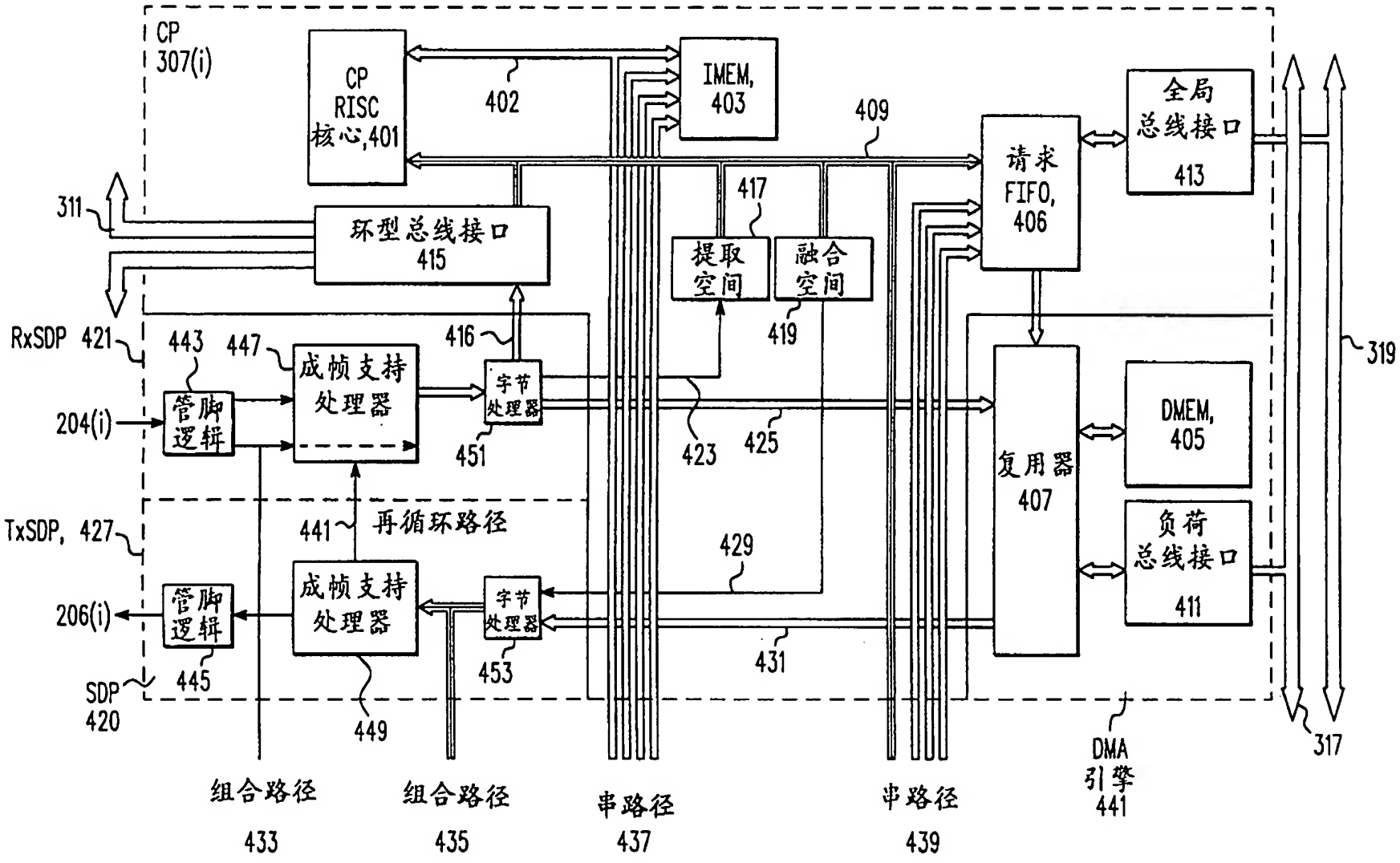
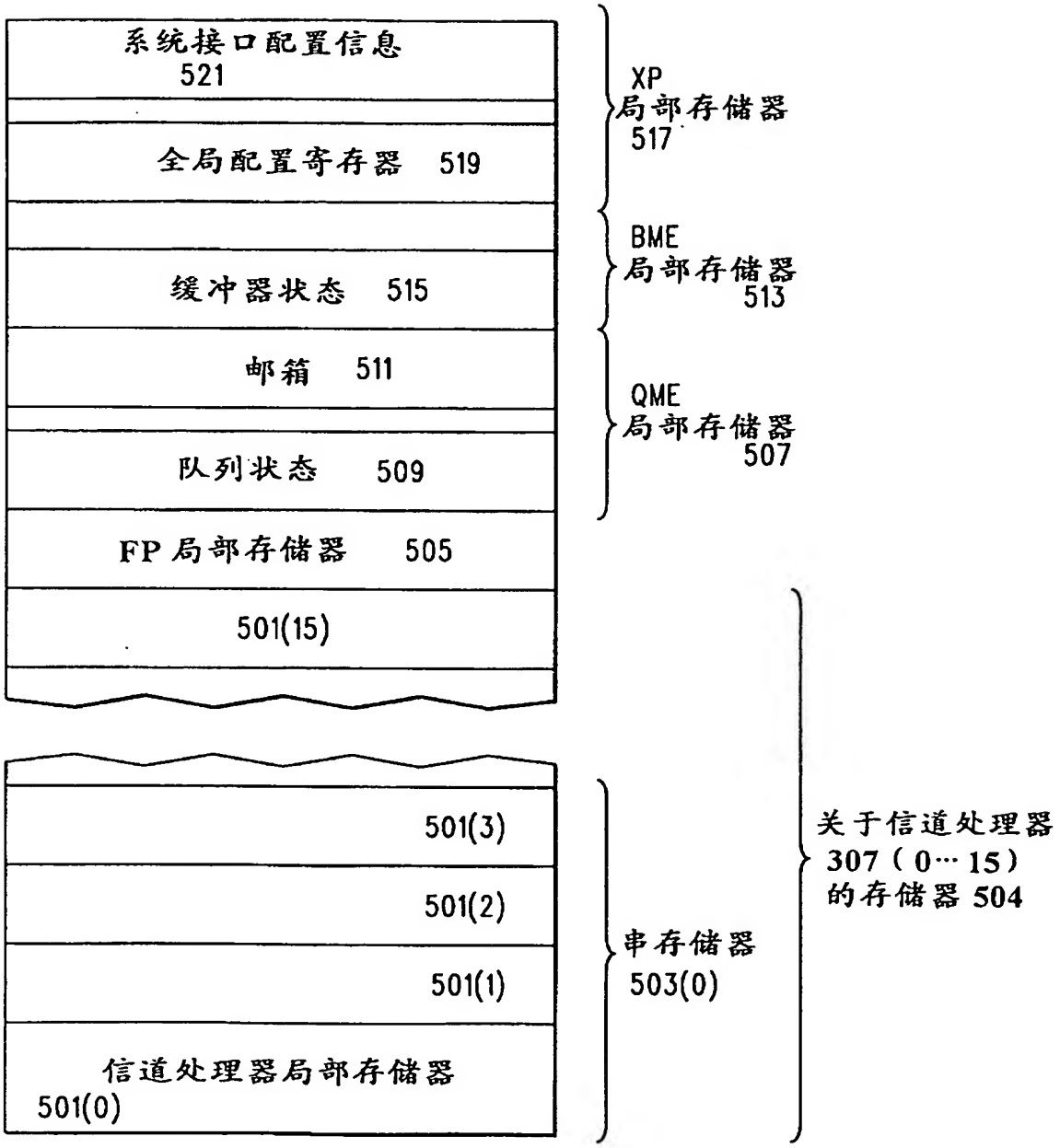

CP
307(i)
CP
RISC
核心,401
402
IMEM,
403
409
全局
总线接口
413
请求
FIFO,
406
311
环型总线接口
415
417
提取
空间
融合
空间
419
RxSDP 421
443
447
416
成帧支持
处理器
字节
处理器
451
423
425
管脚
逻辑
204(i)
复用器
407
DMEM,
405
319
TxSDP, 427
441
再循环路径
429
负荷
总线接口
411
成帧支持
处理器
字节
处理器
453
431
管脚
逻辑
206(i)
445
SDP
420
449
DMA
引擎
441
317
组合路径
433
组合路径
435
串路径
437
串路径
439

图 4


系统接口配置信息
521
全局配置寄存器 519
缓冲器状态 515
邮箱 511
队列状态 509
FP 局部存储器 505
501(15)
501(3)
501(2)
501(1)
信道处理器局部存储器
501(0)
XP
局部存储器
517
BME
局部存储器
513
QME
局部存储器
507
串存储器
503(0)
关于信道处理器
307（0…15）
的存储器 504

321

图 5

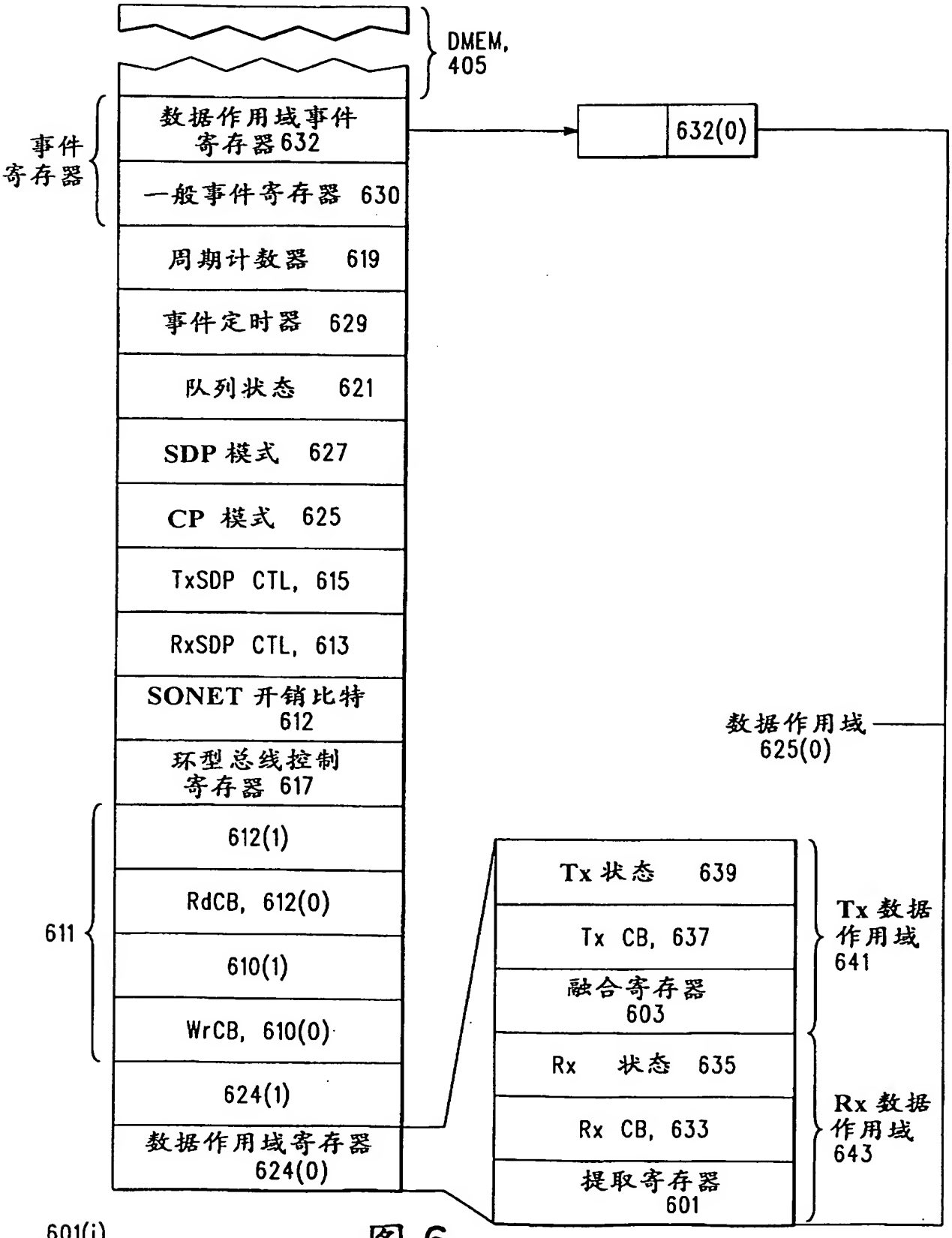

DMEM, 405
事件寄存器
数据作用域事件寄存器632
一般事件寄存器 630
周期计数器 619
事件定时器 629
队列状态 621
SDP 模式 627
CP 模式 625
TxSDP CTL, 615
RxSDP CTL, 613
SONET 开销比特 612
环型总线控制寄存器 617
611
612(1)
RdCB, 612(0)
610(1)
WrCB, 610(0)
624(1)
数据作用域寄存器 624(0)
632(0)
数据作用域 625(0)
Tx 状态 639
Tx CB, 637
融合寄存器 603
Rx 状态 635
Rx CB, 633
提取寄存器 601
Tx 数据作用域 641
Rx 数据作用域 643
601(i)

图 6

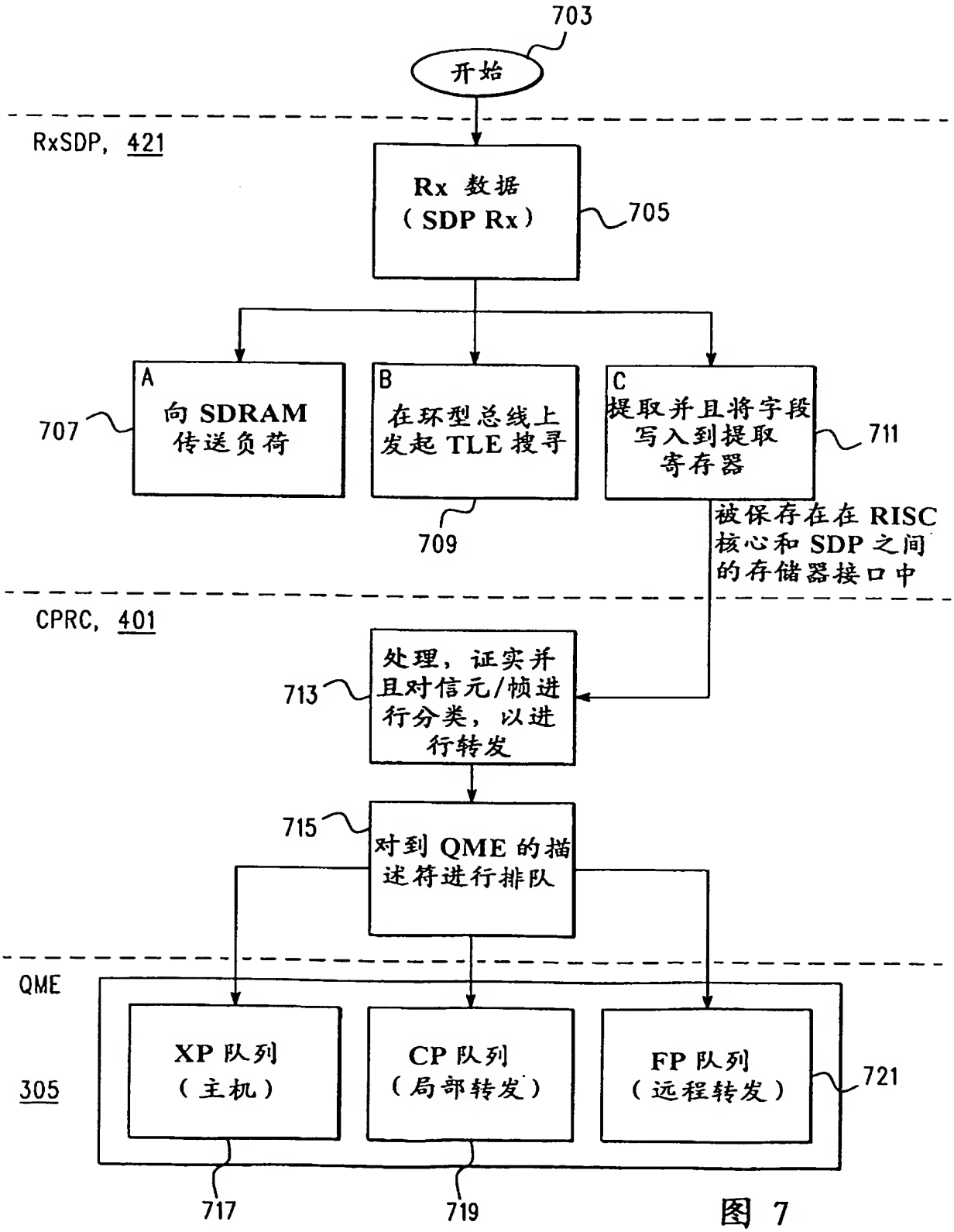

703
开始
RxSDP, 421
Rx 数据
（SDP Rx）
705
A
向 SDRAM
传送负荷
707
B
在环型总线上
发起 TLE 搜寻
709
C
提取并且将字段
写入到提取
寄存器
711
被保存在在 RISC
核心和 SDP 之间
的存储器接口中
CPRC, 401
处理，证实并
且对信元/帧进
行分类，以进
行转发
713
715
对到 QME 的描
述符进行排队
QME
305
XP 队列
（主机）
CP 队列
（局部转发）
FP 队列
（远程转发）
721
717
719
701

图 7

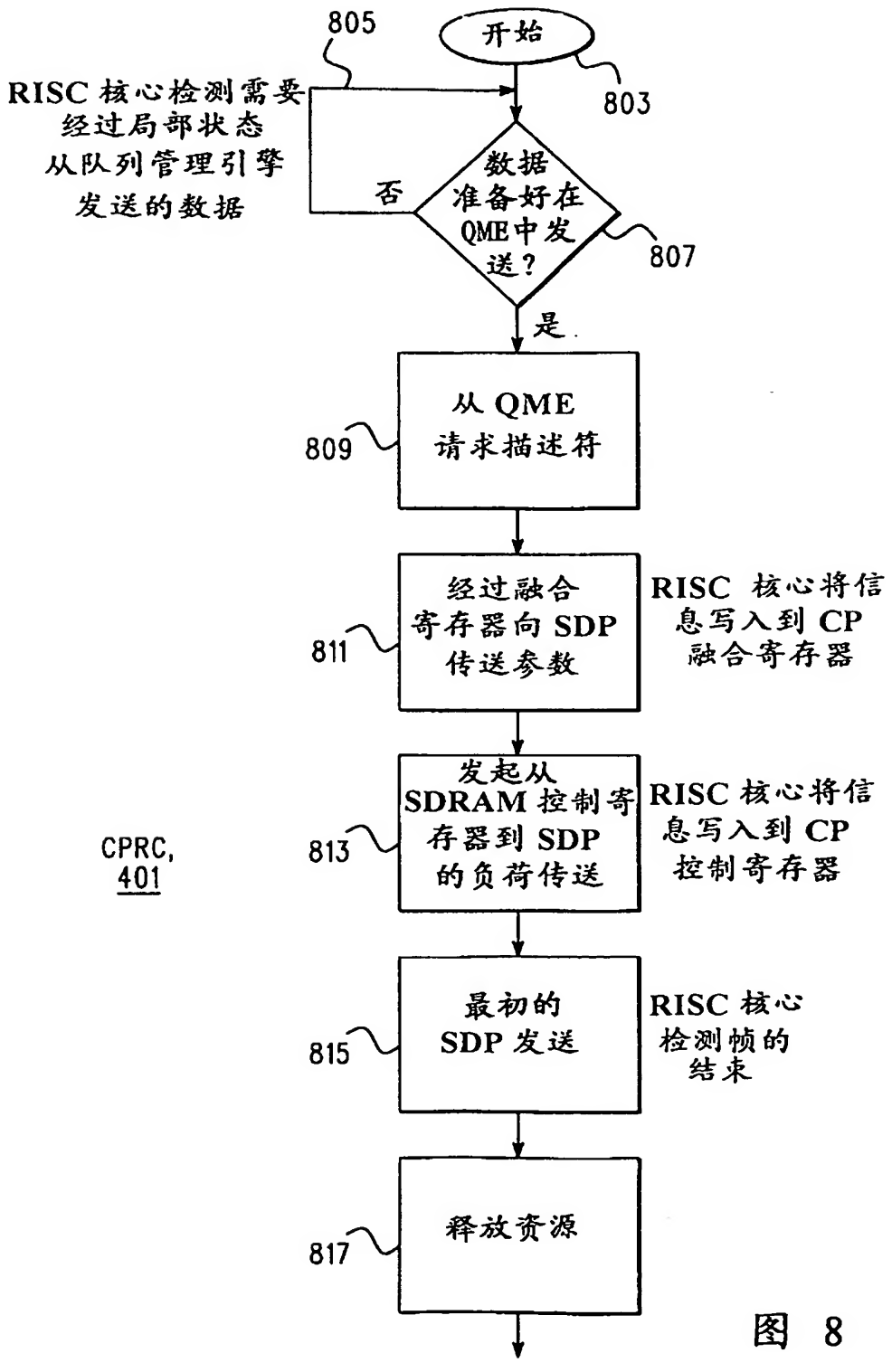

805
开始
803
RISC 核心检测需要经过局部状态从队列管理引擎发送的数据
否
数据准备好在QME中发送?
807
是
从 QME 请求描述符
809
经过融合寄存器向 SDP 传送参数
811
RISC 核心将信息写入到 CP 融合寄存器
发起从 SDRAM 控制寄存器到 SDP 的负荷传送
813
RISC 核心将信息写入到 CP 控制寄存器
CPRC, 401
最初的 SDP 发送
815
RISC 核心检测帧的结束
释放资源
817
801

图 8

| 拥有，935 | L5:L0 937 | 忙 941 |
|---|---|---|

635

| BTAG，933 | | | |
|---|---|---|---|
| 偏移，931 | | | |
| Av，929 | NR，927 | Err，925 | 拥有，921 |
| SDPST，915 | EOP，927 | BCTL，ST 919 | |
| 长度，911 | | | |
| 缓冲器池号码，909 | | | |
| DEM DMA 地址，907 | | | |
| TxRcy 地址，905 | | | |
| RxRcy 地址，903 | | | |
| DMEM 字节地址，901 | | | |

RxCBCTL，913

633

图 9

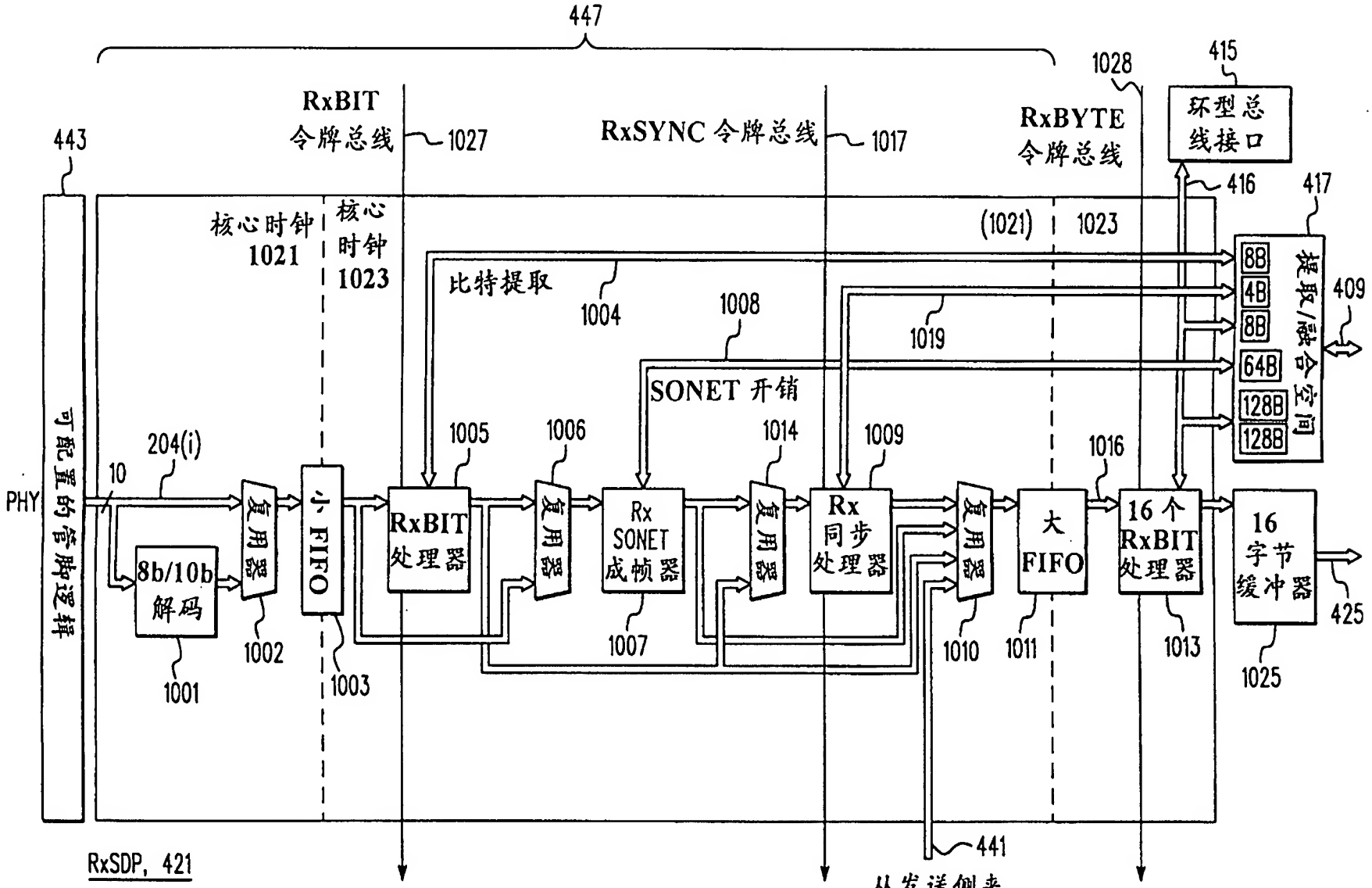

447
RxBIT
令牌总线
1027
RxSYNC 令牌总线
1017
RxBYTE
令牌总线
1028
环型总
线接口
415
443
核心时钟
1021
核心
时钟
1023
(1021)
1023
416
417
比特提取
1004
1008
1019
SONET 开销
409
8B
4B
8B
64B
128B
128B
提取/融合空间
PHY
可配置的管脚逻辑
204(i)
10
8b/10b
解码
1001
复用器
1002
小 FIFO
1003
RxBIT
处理器
1005
复用器
1006
Rx
SONET
成帧器
1007
1014
复用器
Rx
同步
处理器
1009
复用器
1010
大
FIFO
1011
1016
16 个
RxBIT
处理器
1013
16
字节
缓冲器
1025
425
441
从发送侧来
(字节环回)
(再循环)
RxSDP, 421
1002,1006,1014,1010=旁路路径 1015

图 10

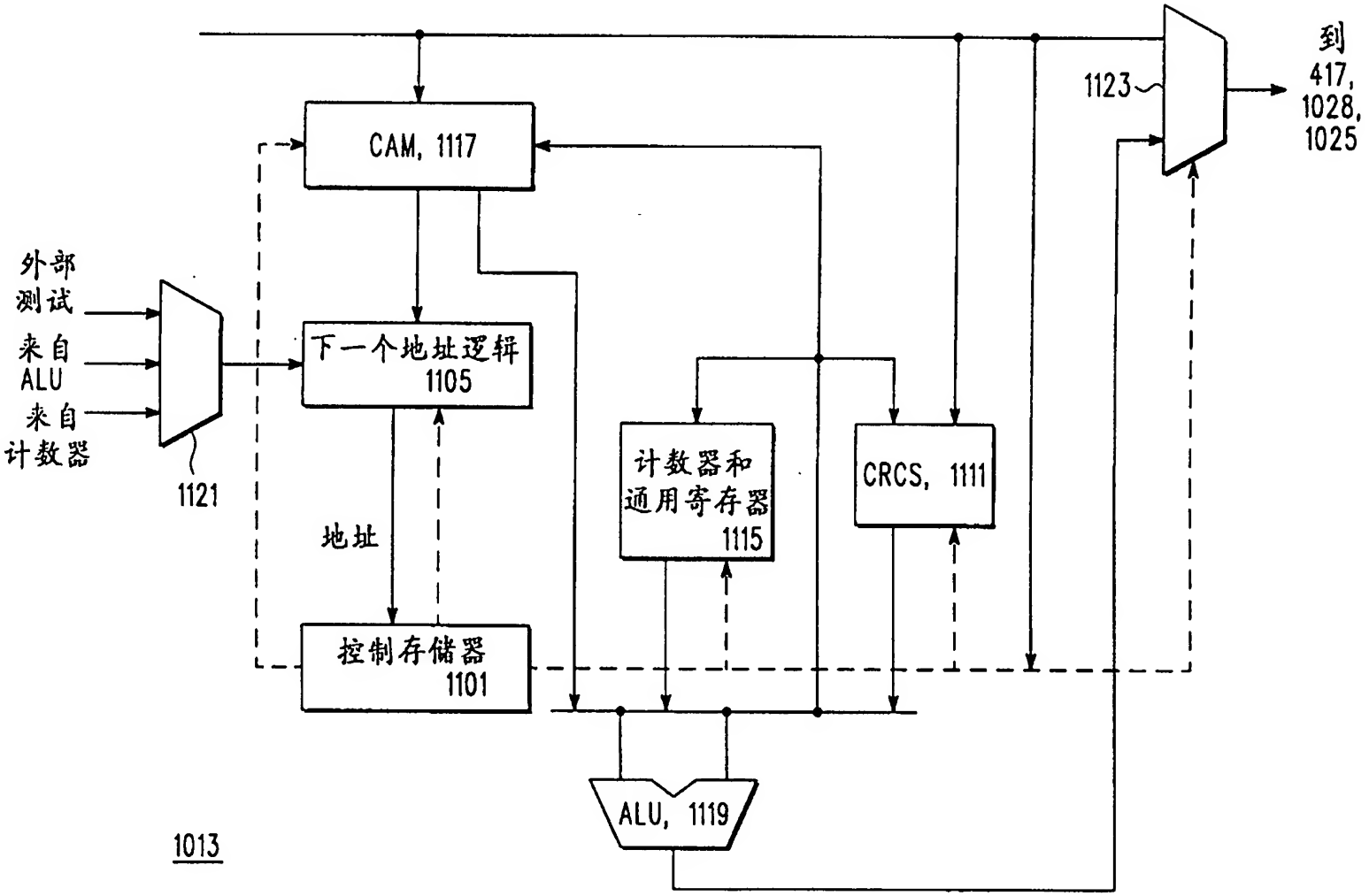

CAM, 1117
外部测试
来自ALU
来自计数器
1121
下一个地址逻辑 1105
地址
控制存储器 1101
计数器和通用寄存器 1115
CRCS, 1111
ALU, 1119
1123
到 417, 1028, 1025
1013

图 11

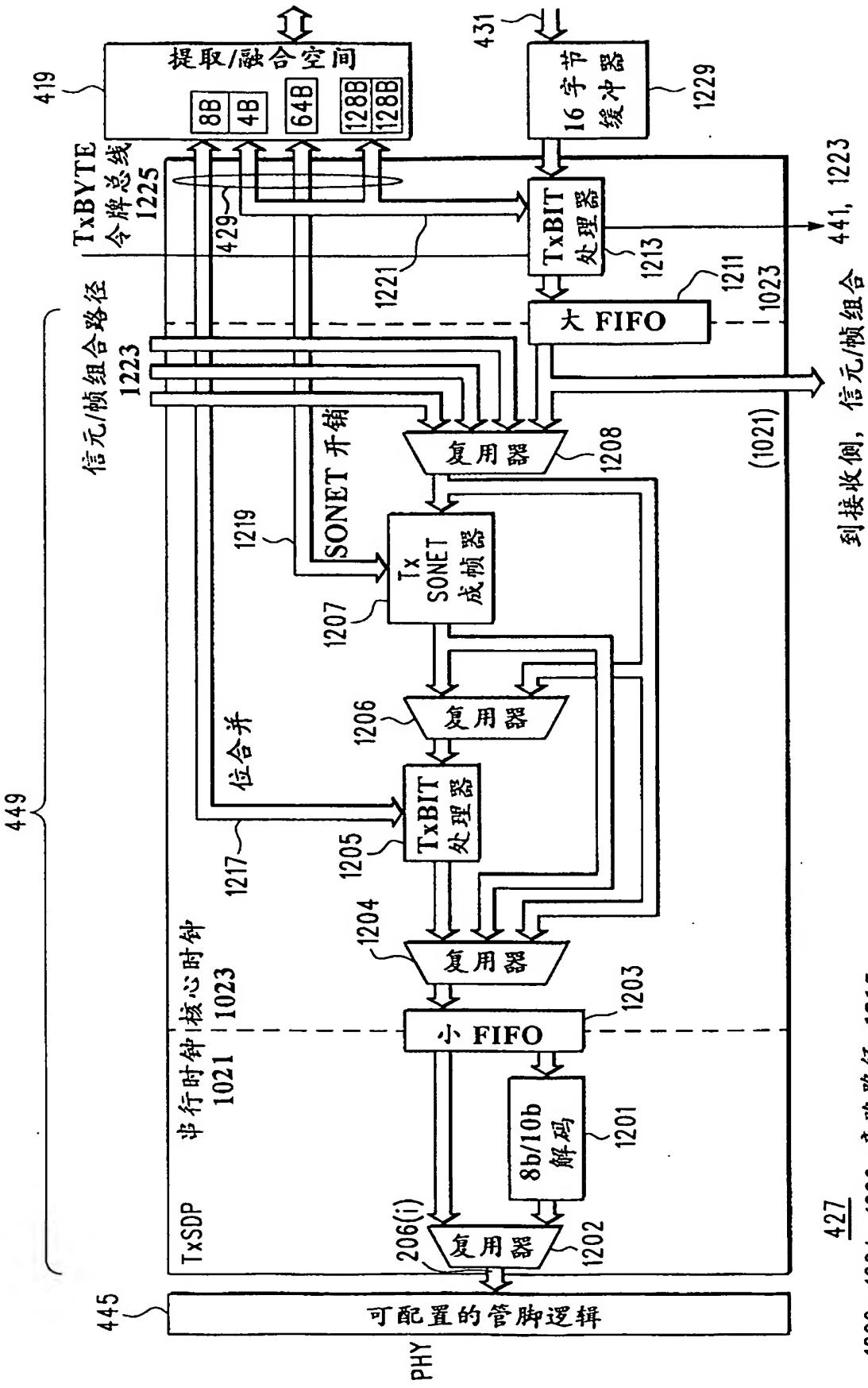

提取/融合空间
8B
4B
64B
128B
128B
419
431
16 字节缓冲器
1229
TxBYTE 令牌总线 1225
429
TxBIT 处理器
1213
441, 1223
1221
1211
1023
大 FIFO
信元/帧组合路径 1223
复用器
1208
SONET 开销
1219
Tx SONET 成帧器
1207
(1021)
到接收侧，信元/帧组合（字节环回）和 Tx 组合
复用器
1206
位合并
TxBIT 处理器
1205
1217
449
1204
复用器
核心时钟 1023
1203
小 FIFO
串行时钟 1021
8b/10b 解码
1201
TxSDP
206(i)
复用器
1202
427
1202, 1204, 1206, 旁路路径 1215
445
可配置的管脚逻辑
PHY

图 12

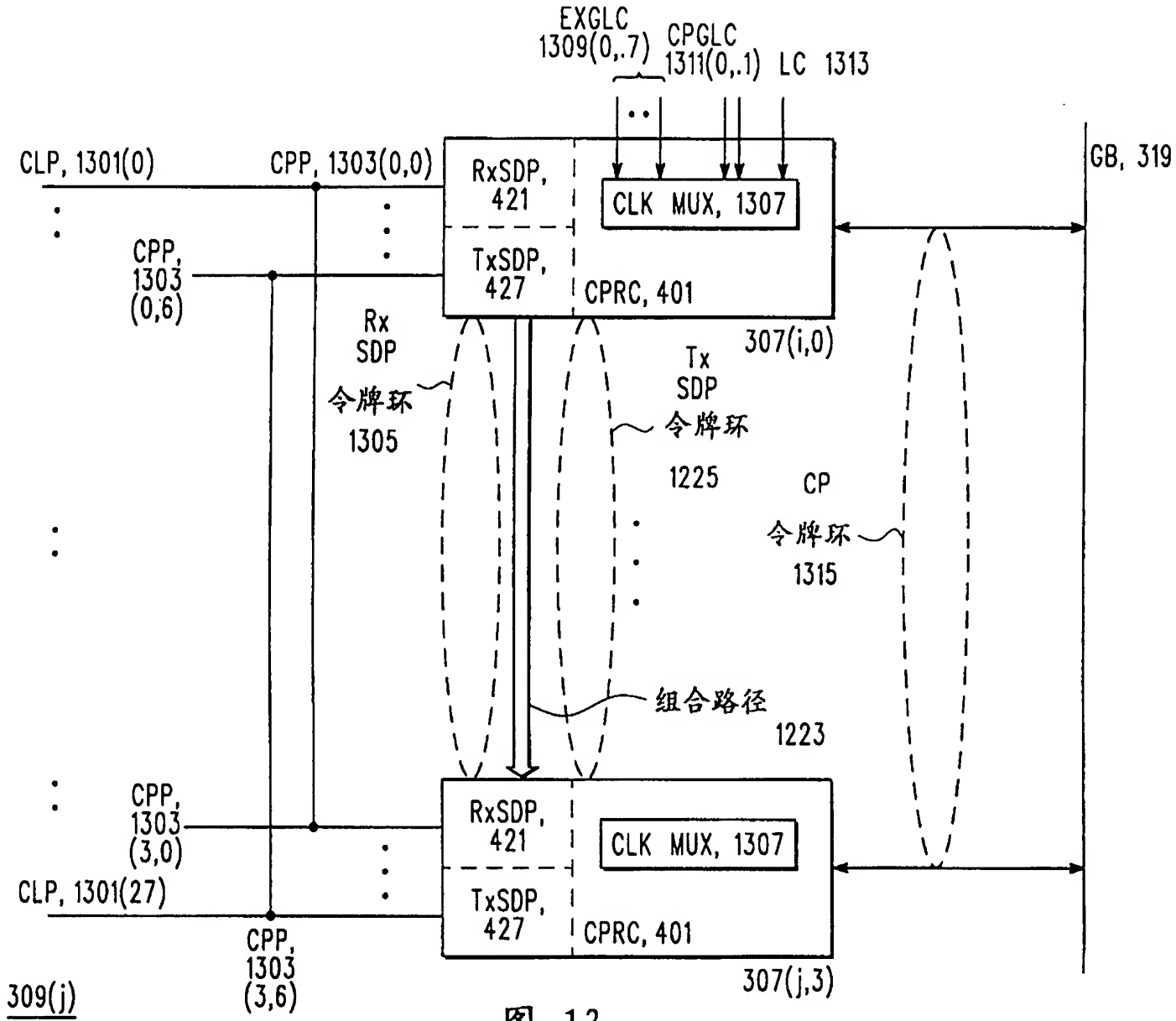

EXGLC
1309(0,.7)
CPGLC
1311(0,.1)
LC 1313
CLP, 1301(0)
CPP, 1303(0,0)
RxSDP,
421
CLK MUX, 1307
GB, 319
CPP,
1303
(0,6)
TxSDP,
427
CPRC, 401
307(i,0)
Rx
SDP
令牌环
1305
Tx
SDP
令牌环
1225
CP
令牌环
1315
组合路径
1223
CPP,
1303
(3,0)
RxSDP,
421
CLK MUX, 1307
CLP, 1301(27)
TxSDP,
427
CPRC, 401
CPP,
1303
(3,6)
307(j,3)
309(j)

图 13

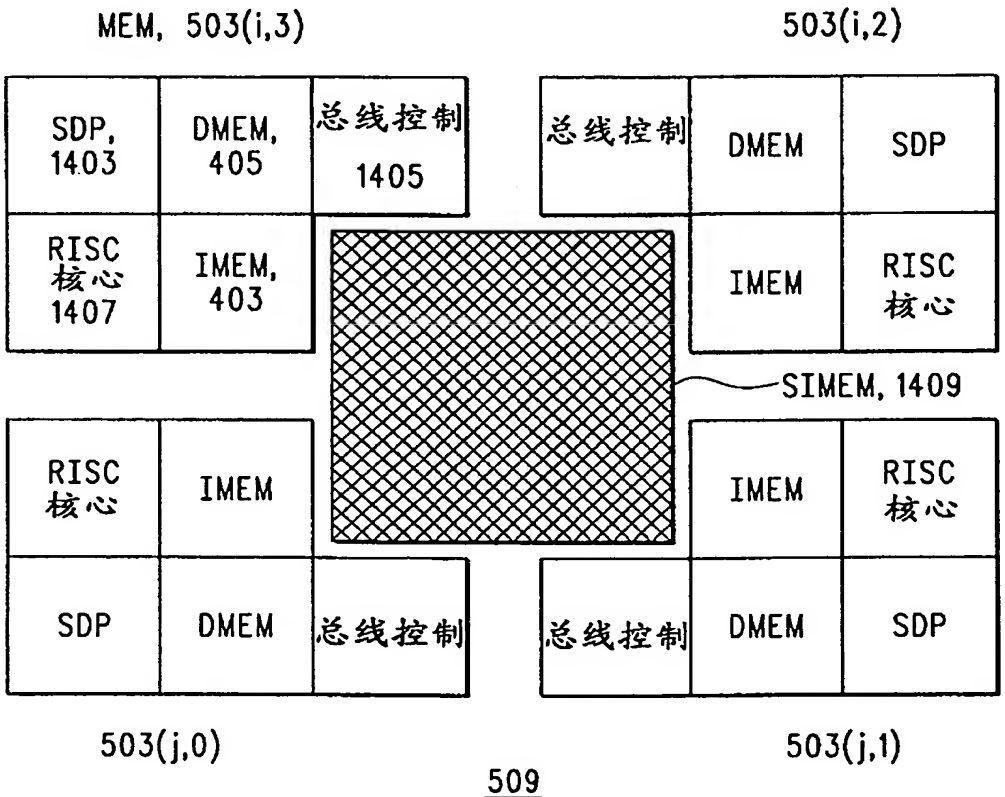

MEM, 503(i,3)
503(i,2)
SDP, 1403
DMEM, 405
总线控制 1405
总线控制
DMEM
SDP
RISC 核心 1407
IMEM, 403
IMEM
RISC 核心
SIMEM, 1409
RISC 核心
IMEM
IMEM
RISC 核心
SDP
DMEM
总线控制
总线控制
DMEM
SDP
503(j,0)
503(j,1)
509

图 14

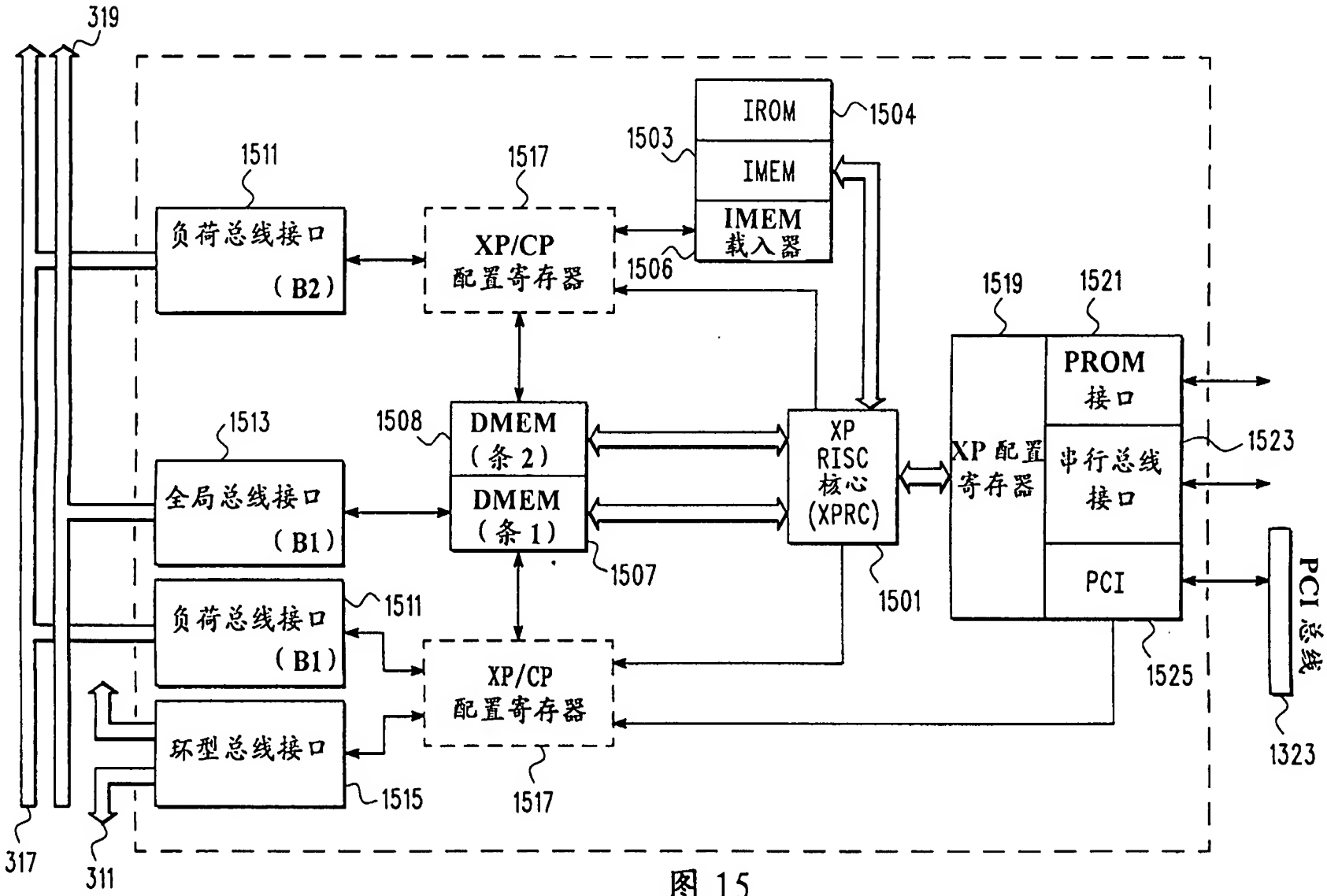

319
IROM
1504
1503
IMEM
1511
1517
负荷总线接口
(B2)
XP/CP
配置寄存器
IMEM
载入器
1506
1519
1521
PROM
接口
1513
1508
DMEM
(条2)
XP
RISC
核心
(XPRC)
XP 配置
寄存器
串行总线
接口
1523
全局总线接口
(B1)
DMEM
(条1)
1507
1501
PCI
PCI 总线
负荷总线接口
(B1)
1511
XP/CP
配置寄存器
1525
环型总线接口
1515
1517
1323
317
311

图 15

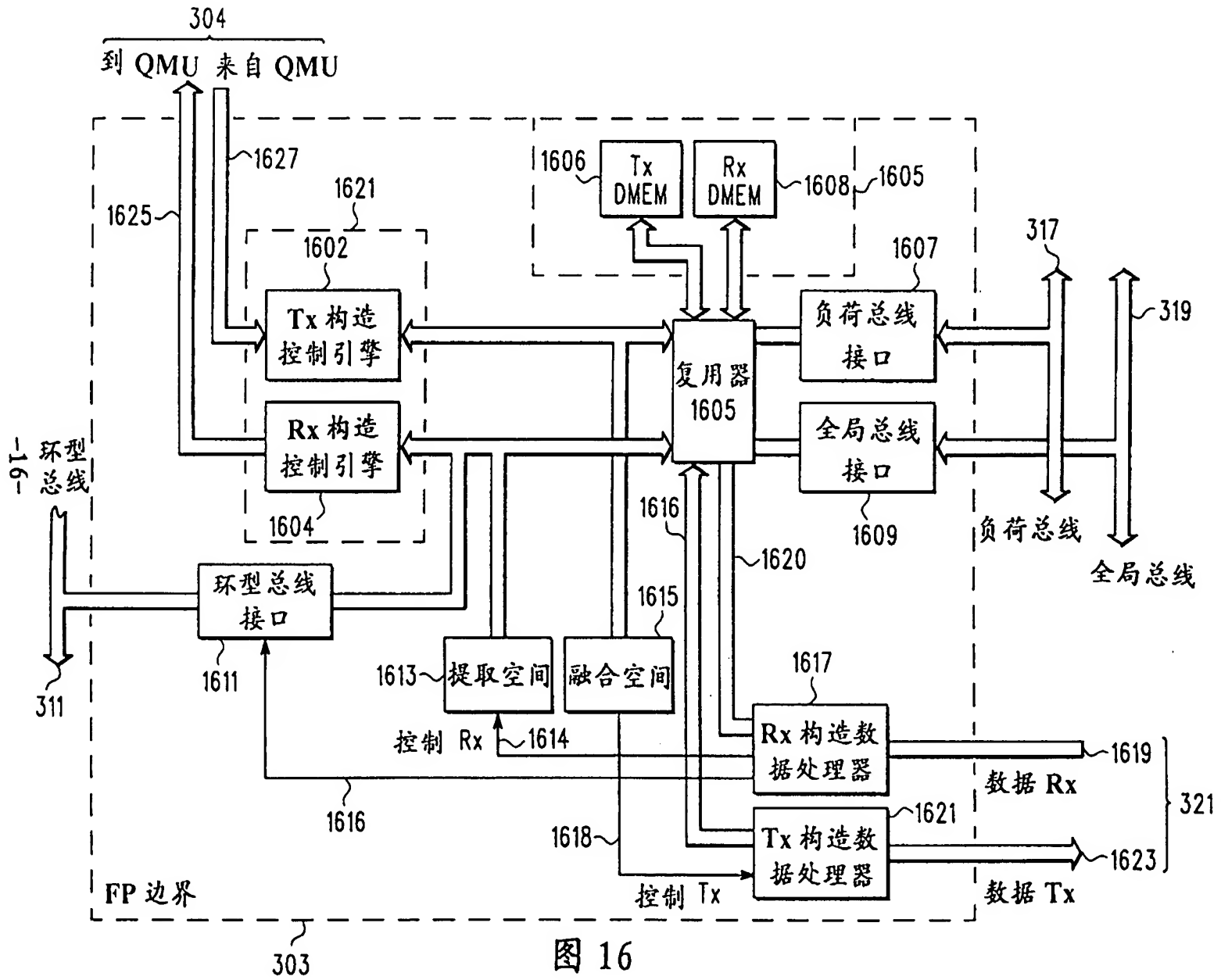

304
到 QMU
来自 QMU
1627
1625
1621
1602
1606
Tx DMEM
Rx DMEM
1608
1605
1607
317
319
Tx 构造控制引擎
Rx 构造控制引擎
1604
复用器 1605
负荷总线接口
全局总线接口
1609
负荷总线
全局总线
环型总线
-16-
环型总线接口
1611
311
1616
1620
1615
1613
提取空间
融合空间
1617
控制 Rx
1614
Rx 构造数据处理器
1619
数据 Rx
321
1621
Tx 构造数据处理器
1623
数据 Tx
1618
控制 Tx
FP 边界
303

图 16

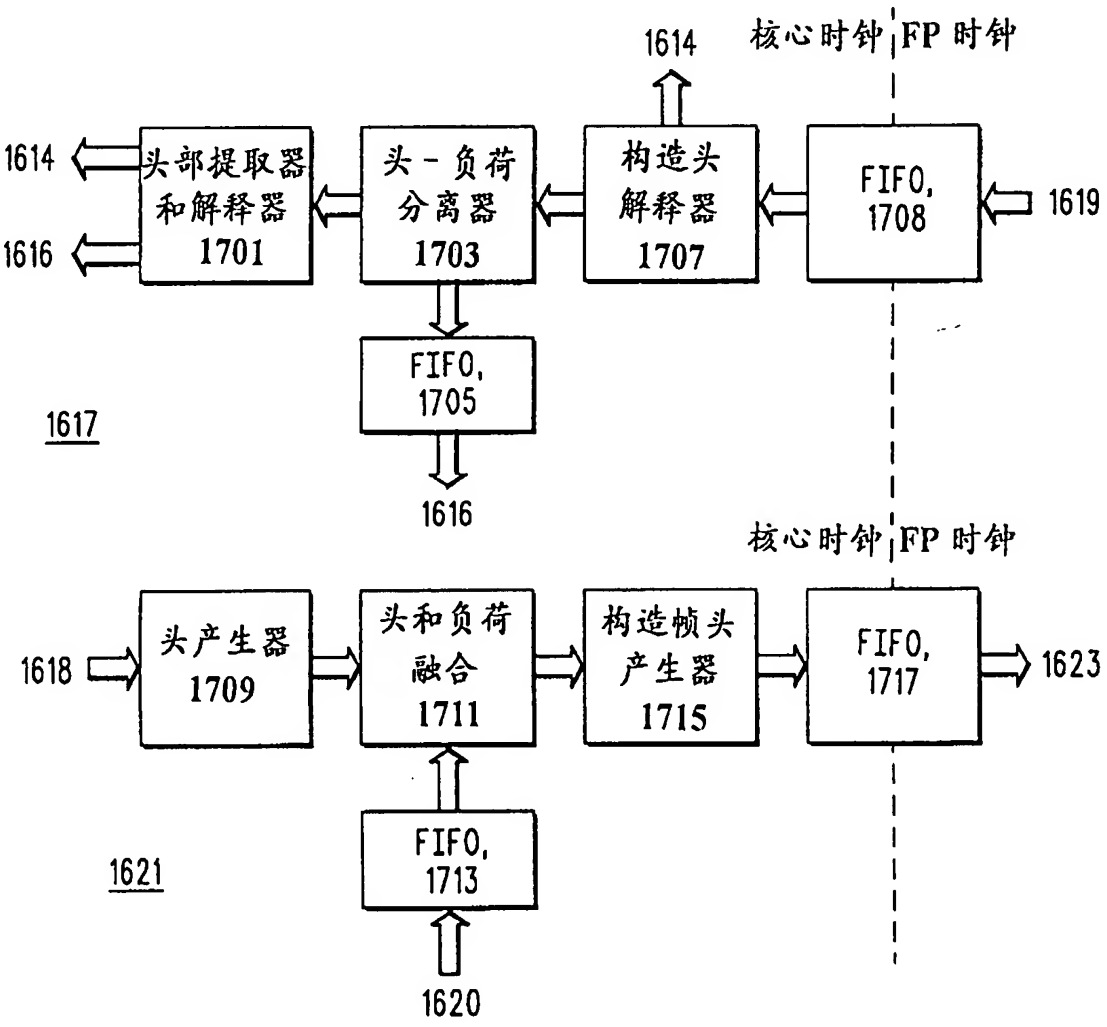

1614
核心时钟
FP 时钟
1614
头部提取器和解释器 1701
头－负荷分离器 1703
构造头解释器 1707
FIFO, 1708
1619
1616
FIFO, 1705
1617
1616
核心时钟
FP 时钟
1618
头产生器 1709
头和负荷融合 1711
构造帧头产生器 1715
FIFO, 1717
1623
FIFO, 1713
1621
1620

图 17

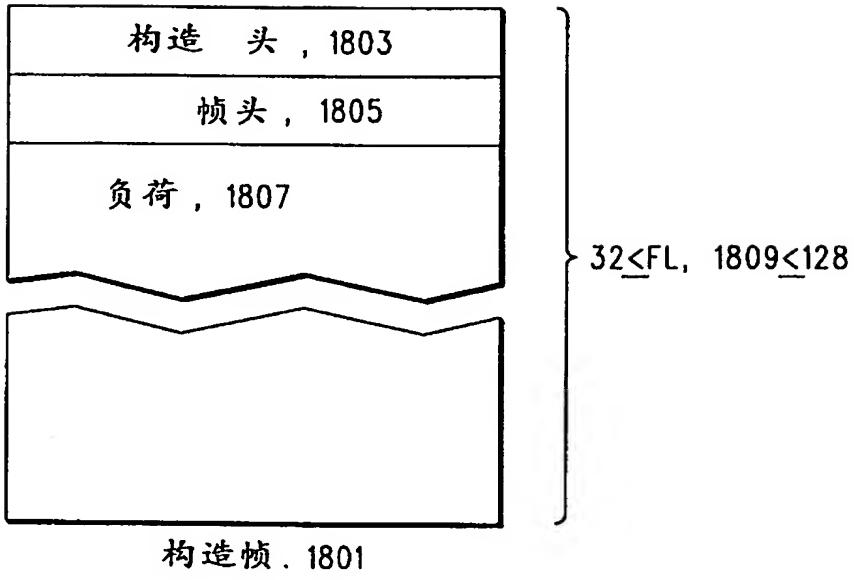



图 18

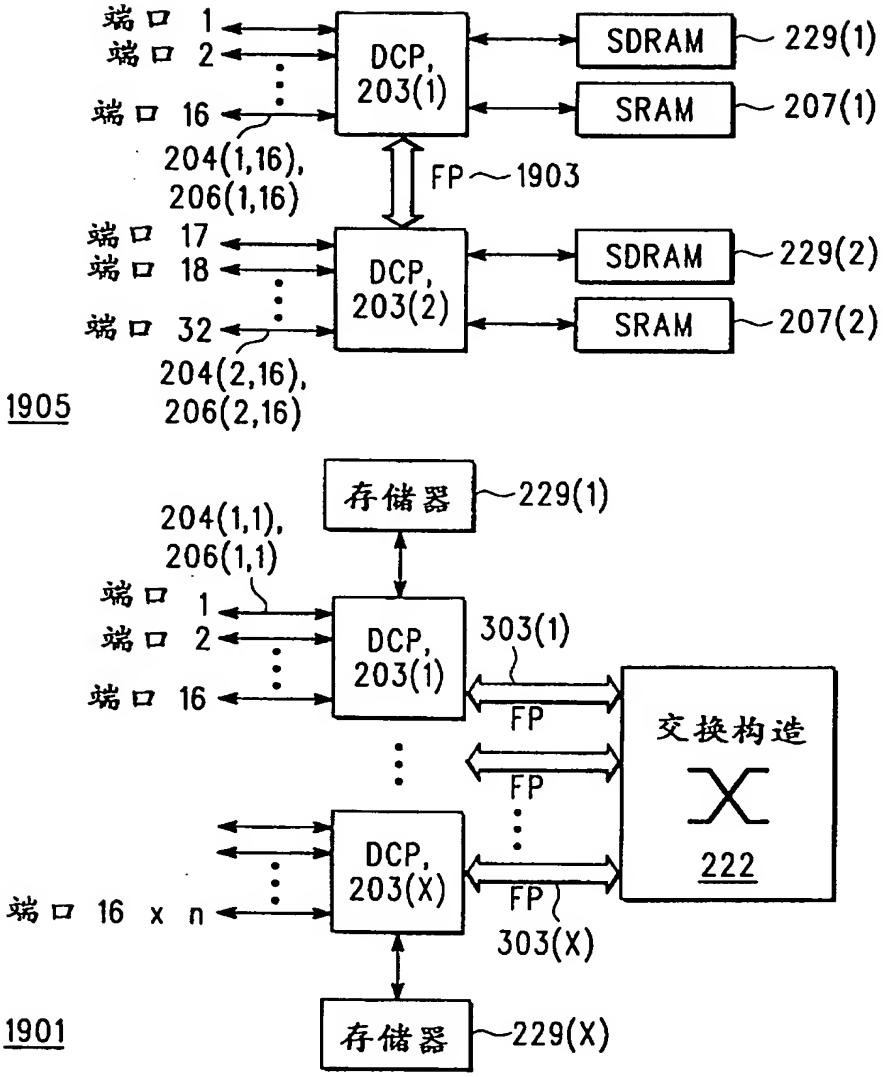

端口 1
端口 2
端口 16
DCP, 203(1)
SDRAM
229(1)
SRAM
207(1)
204(1,16), 206(1,16)
FP
1903
端口 17
端口 18
端口 32
DCP, 203(2)
SDRAM
229(2)
SRAM
207(2)
204(2,16), 206(2,16)
1905
存储器
229(1)
204(1,1), 206(1,1)
端口 1
端口 2
端口 16
DCP, 203(1)
303(1)
FP
交换构造
X
222
FP
DCP, 203(X)
端口 16 x n
FP
303(X)
1901
存储器
229(X)

图 19

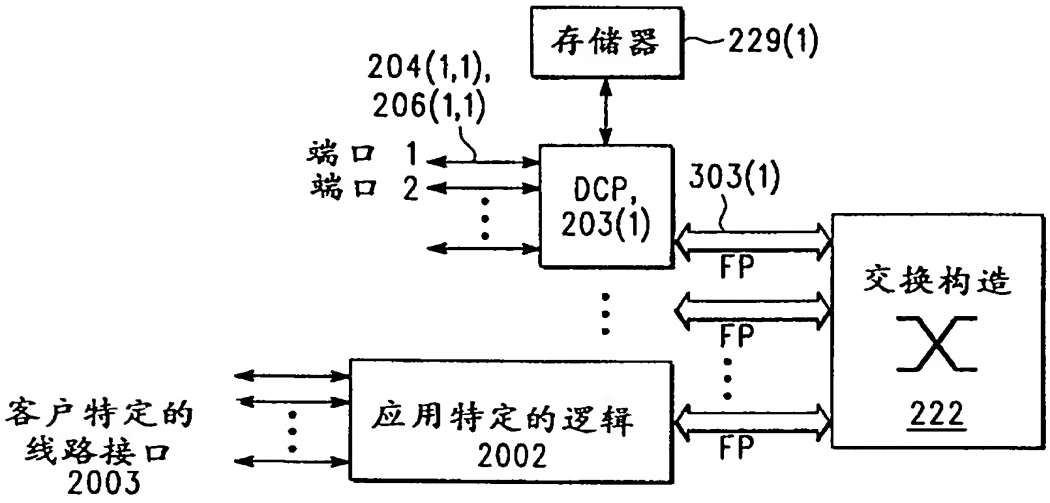

存储器
229(1)
204(1,1),
206(1,1)
端口 1
端口 2
DCP,
203(1)
303(1)
FP
FP
FP
交换构造
X
222
客户特定的
线路接口
2003
应用特定的逻辑
2002
2001

图 20

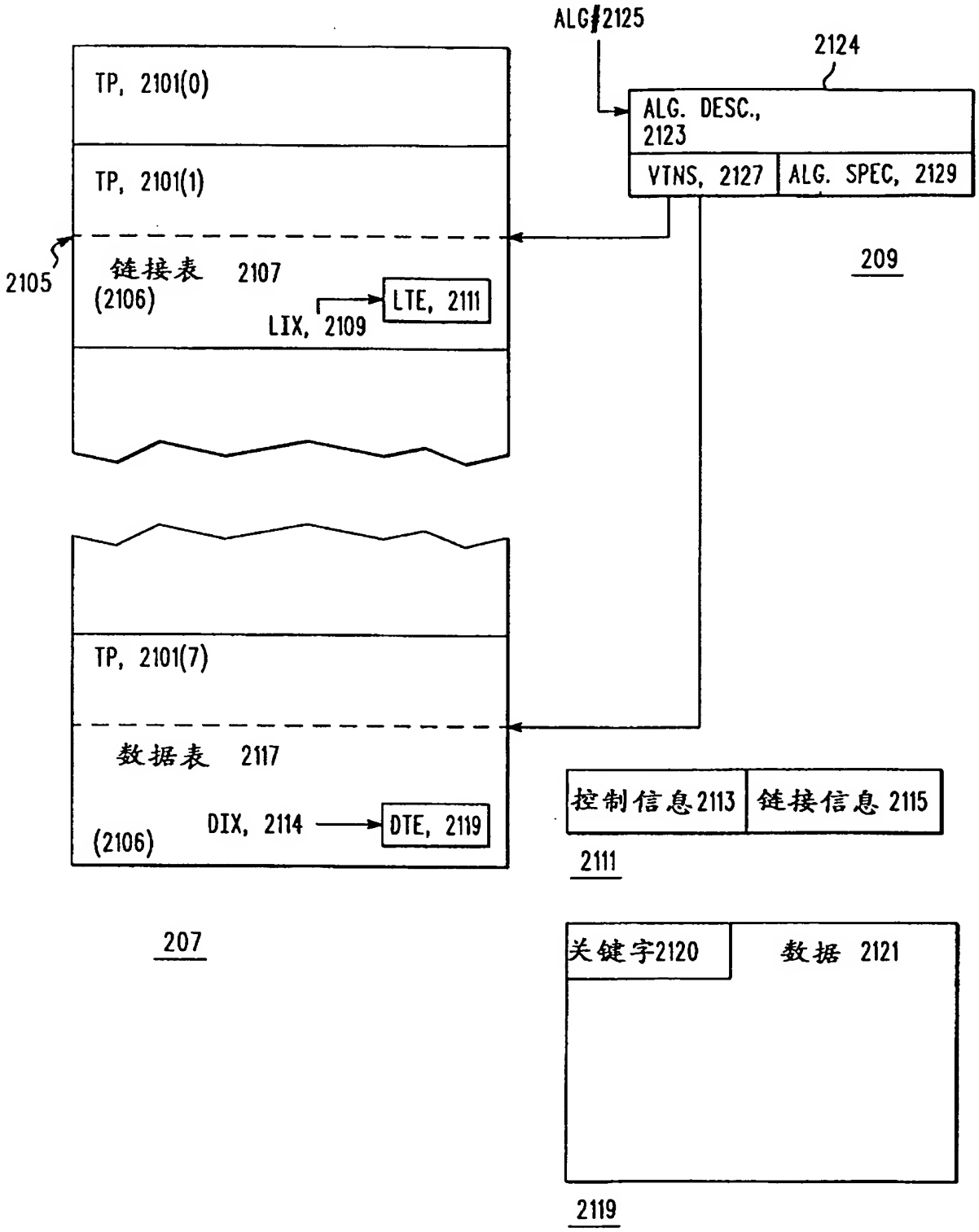

ALG 2125
2124
TP, 2101(0)
TP, 2101(1)
ALG. DESC., 2123
VTNS, 2127
ALG. SPEC, 2129
2105
链接表 2107
(2106)
LIX, 2109
LTE, 2111
209
TP, 2101(7)
数据表 2117
DIX, 2114
DTE, 2119
(2106)
207
控制信息2113
链接信息 2115
2111
关键字2120
数据 2121
2119

图 21

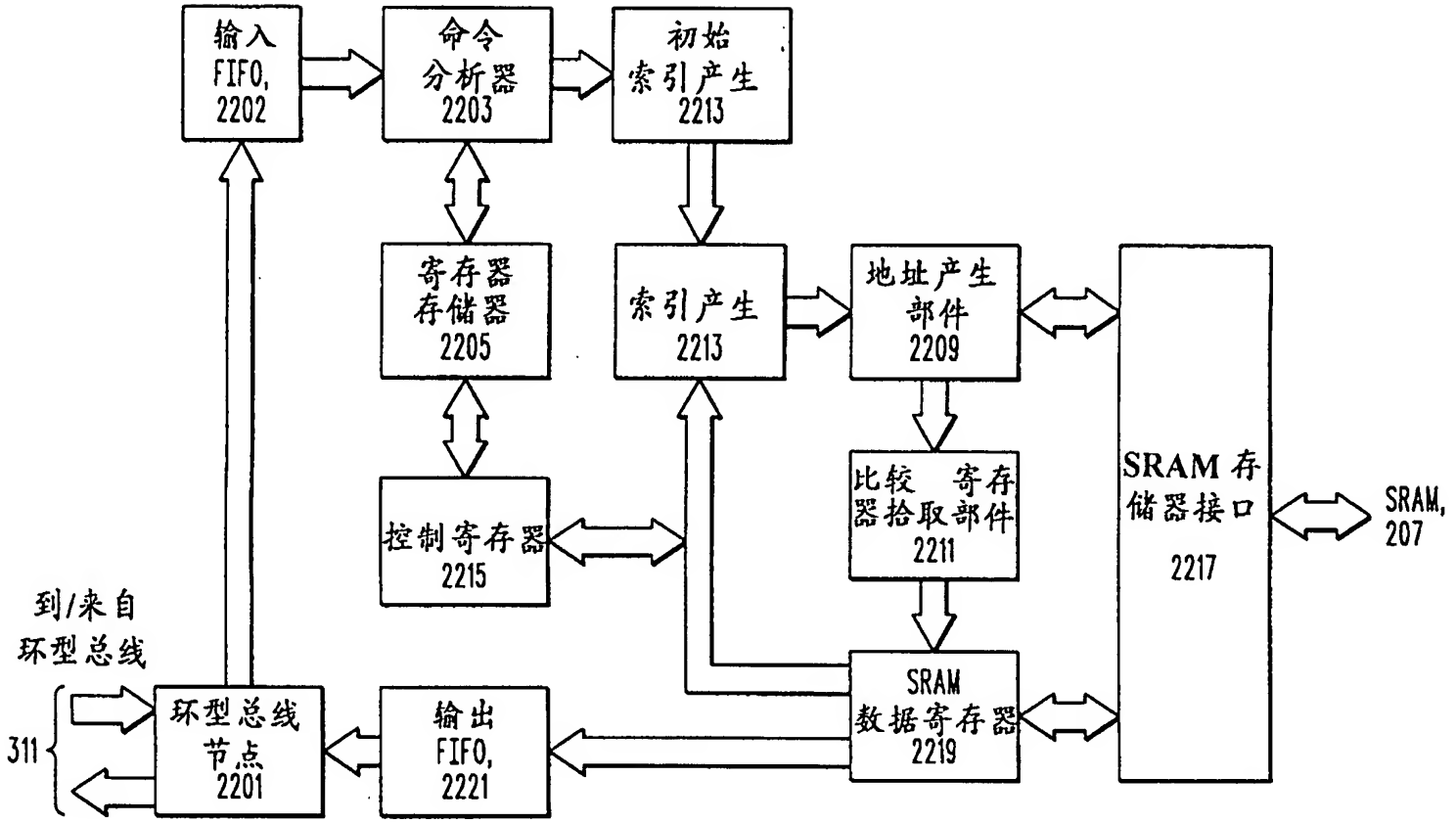

输入
FIFO,
2202
命令
分析器
2203
初始
索引产生
2213
寄存器
存储器
2205
索引产生
2213
地址产生
部件
2209
SRAM 存
储器接口
2217
SRAM,
207
比较 寄存
器拾取部件
2211
控制寄存器
2215
到/来自
环型总线
311
环型总线
节点
2201
输出
FIFO,
2221
SRAM
数据寄存器
2219
301

图 22

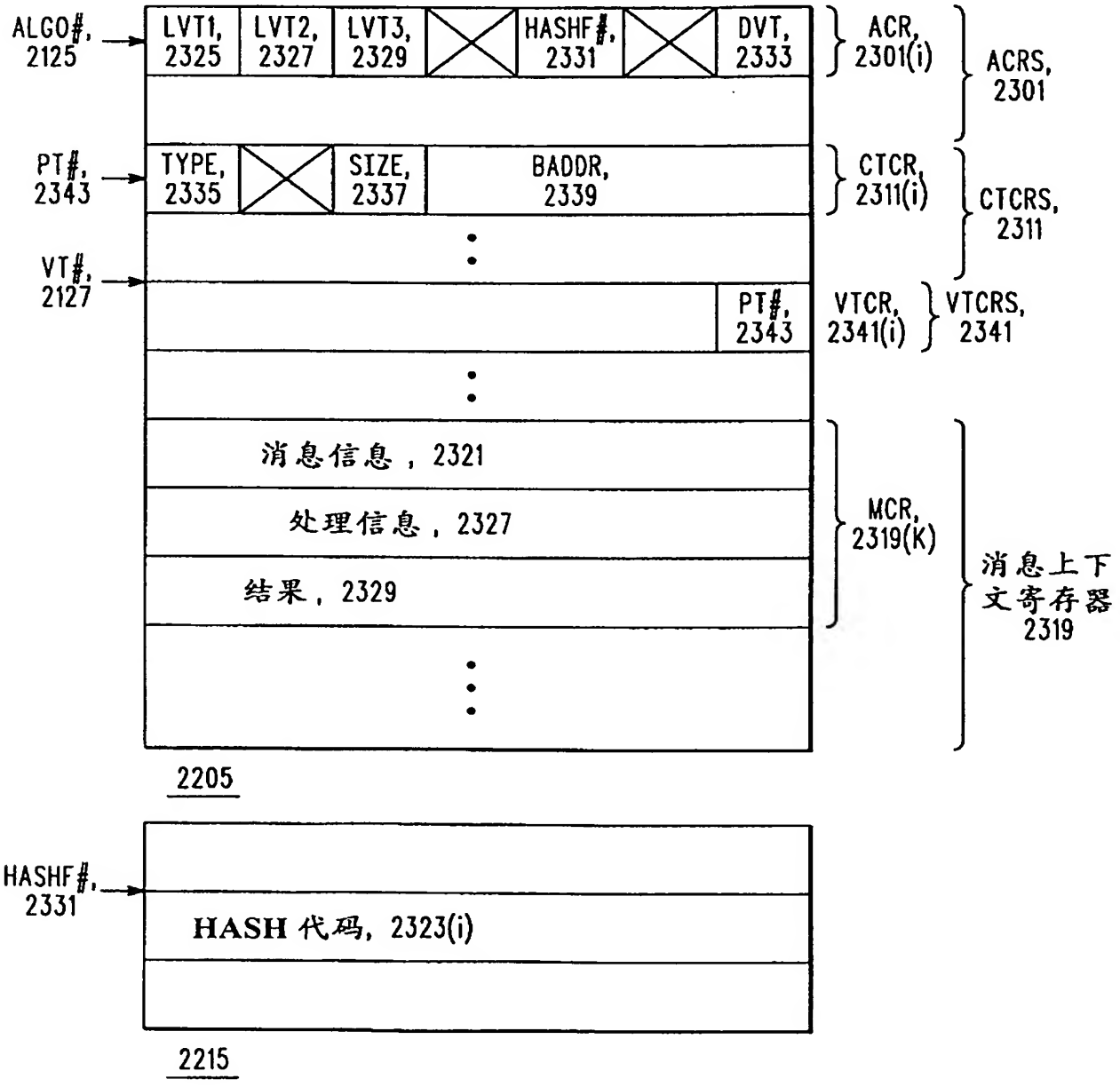

ALGO#, 2125
LVT1, 2325
LVT2, 2327
LVT3, 2329
HASHF#, 2331
DVT, 2333
ACR, 2301(i)
ACRS, 2301
PT#, 2343
TYPE, 2335
SIZE, 2337
BADDR, 2339
CTCR, 2311(i)
CTCRS, 2311
VT#, 2127
PT#, 2343
VTCR, 2341(i)
VTCRS, 2341
消息信息，2321
处理信息，2327
结果，2329
MCR, 2319(K)
消息上下文寄存器 2319
2205
HASHF#, 2331
HASH 代码，2323(i)
2215

图 23

| | 命令 | 命令 ID | 结果数据 | 描述 | |
|---|---|---|---|---|---|
| 2421 | 写入(VTABLE#,索引,掩码,数据,偏移,长度) | 0X2 | 无 | 将数据写入到在索引处的一个虚拟表 | 2401 |
| 2421 | 读取(VTABLE#,索引,偏移,长度) | 0X3 | 数据 | 从一个虚拟表读取数据 | 2403 |
| 2423 | FINDW(ALG#,关键字) | 0X6 | 物理表,索引,错误 | 使用 ALG # 来设置环型总线错误标记,如果关键字没有被发现 | 2405 |
| 2423 | FINDW(ALG#,关键字,数据,偏移,长度) | 0X4 | 通过/失败,索引,错误 | 使用一个关键字将数据写入到一个表. 如果没有发现这个关键字,就设置环型总线错误标记 | 2407 |
| 2423 | FINDR(ALG#,关键字,数据,偏移,长度) | 0X5 | 通过/失败,索引,数据 | 从一个 VTABLE # 读取长为该长度的数据的字 | 2409 |
| 2425 | XOR(VTABLE#,索引,数据/PCRC,偏移,掩码,CRC,最后) | 0X1 | 没有,或者在 CRC 模式下是 CRC | 对偏移的一个32 比特值进行XOR. 仅 4 个连续字节的掩码是有效的. 存在一个特殊的模式来进行 CRC 计算 | 2411 |
| 2425 | ADD(VTABLE#,索引,数据,偏移,掩码) | 0X7 | 无 | 将一个 32 - 比特值添加到偏移处. 仅 4 个连续字节的掩码是有效的. | 2413 |
| 2427 | WRITEREG(REG-ADDR,DATA) | 0X0,0x10 | 无 | 将数据写入到在 REG_ADDR处的 TLE 寄存器 | 2415 |
| 2427 | READREG(REG_ADDR,数据) | 0X0,0x11 | 数据 | 从在 REG_ADDR处的 TLE 寄存器读取数据 | 2417 |
| | ECHO(DATA) | 0X0,0x04 | 数据 | 从 TLE 返回数据,以进行测试 | 2419 |
| | NOP ( ) | 0X0,0x05 | 无 | 在 TLE 流水线中插入一个空操作 | 2420 |

图 24

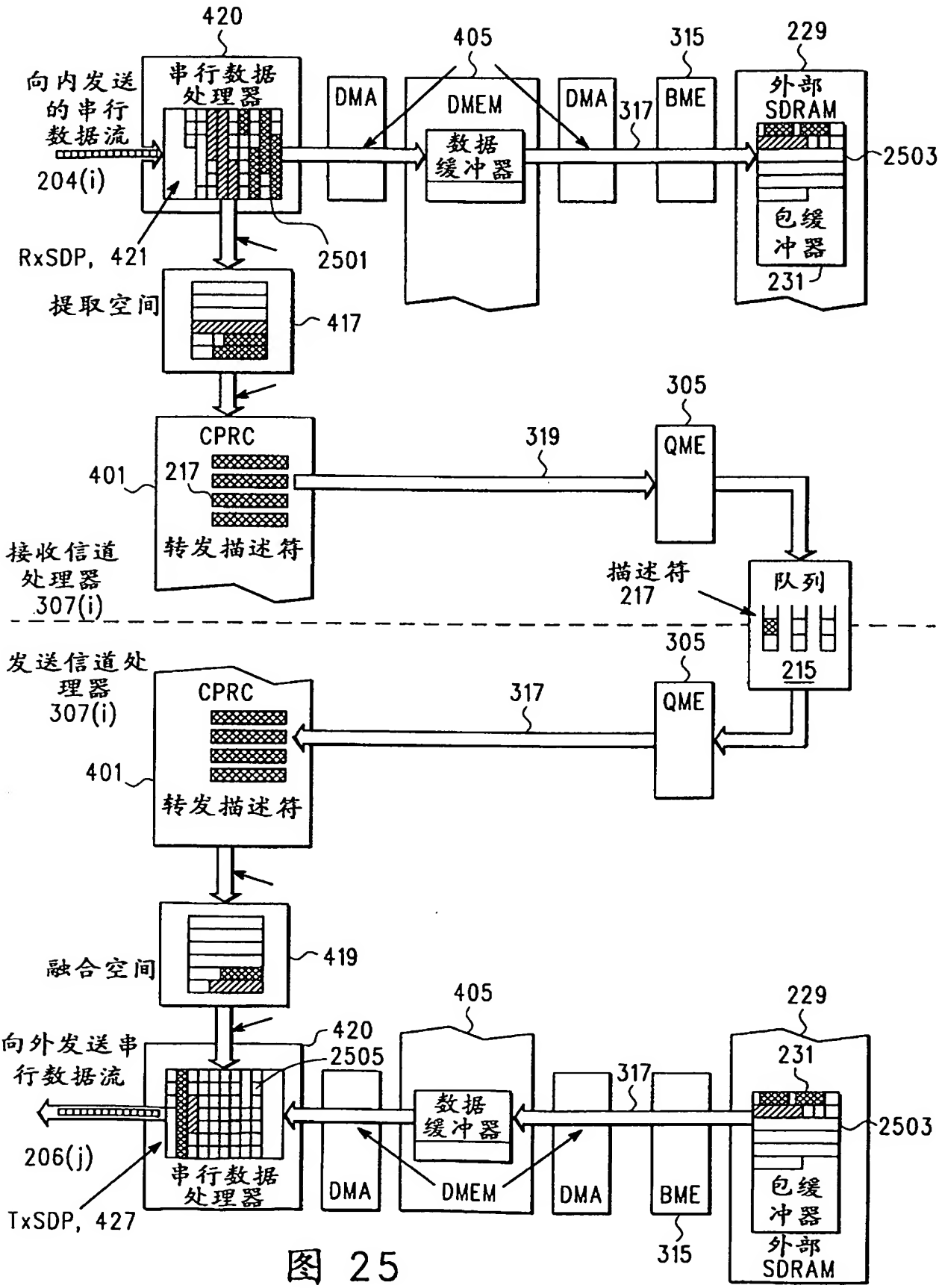
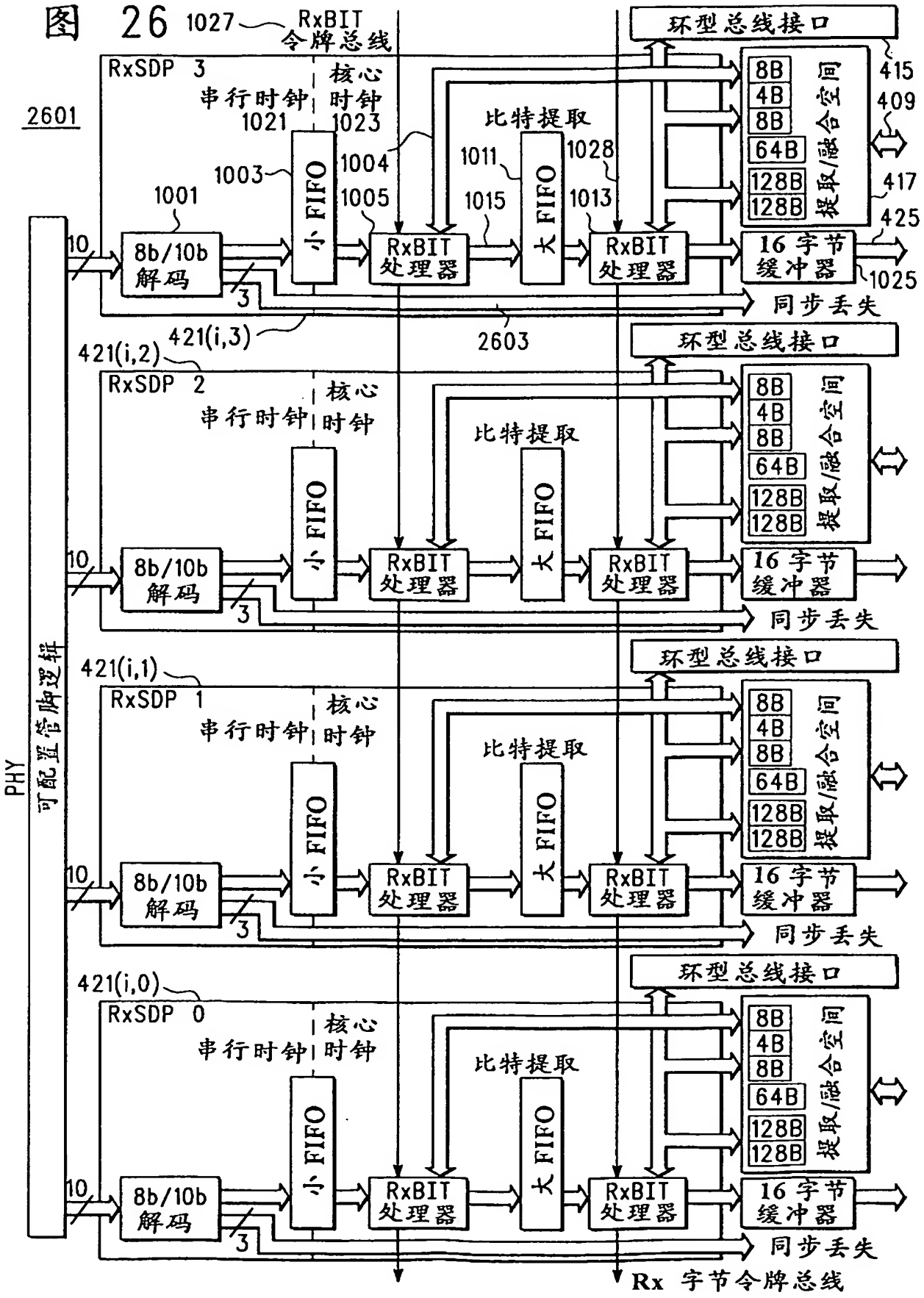
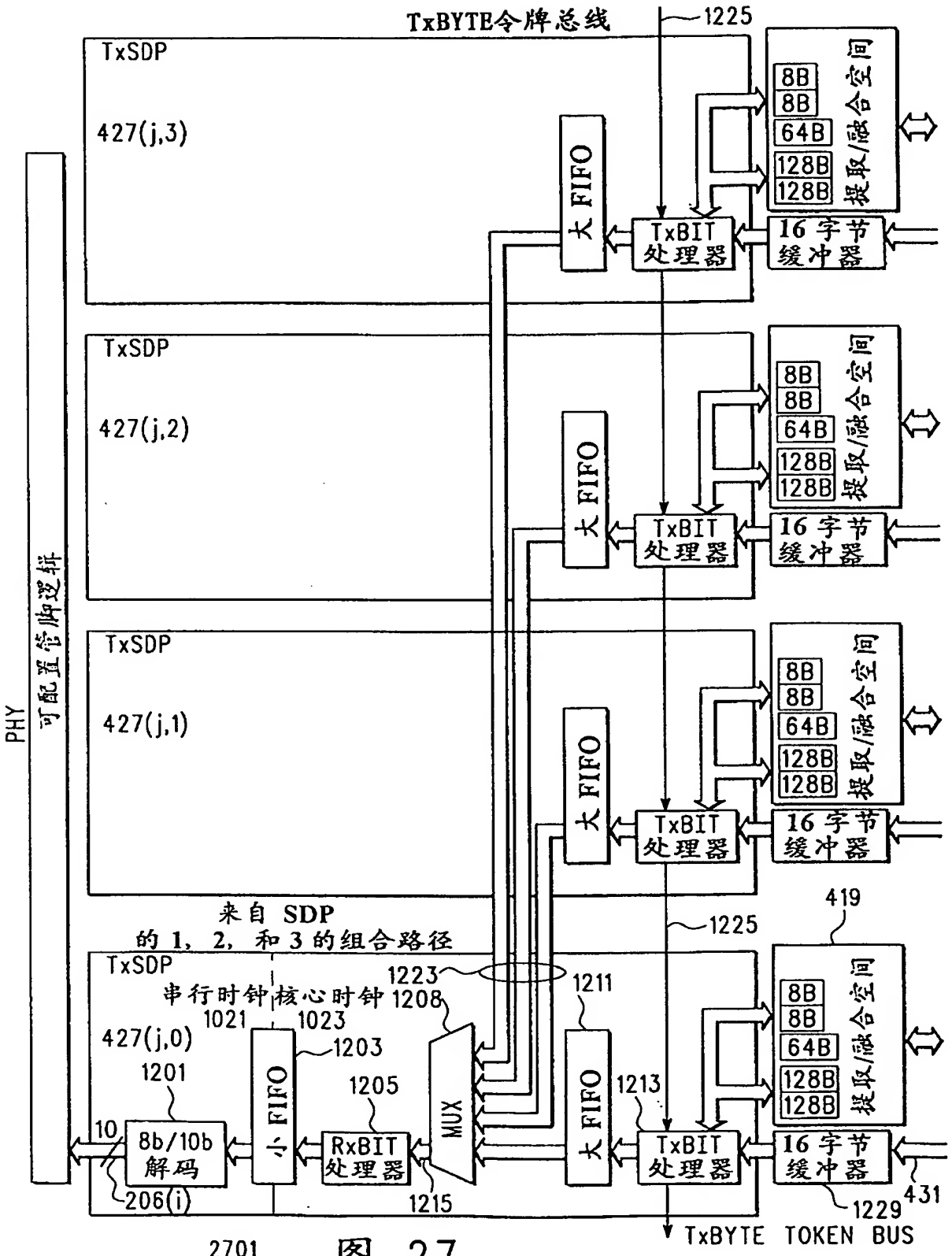

420
405
315
229
向内发送的串行数据流
串行数据处理器
DMA
DMEM
DMA
317
BME
外部SDRAM
数据缓冲器
2503
204(i)
包缓冲器
231
RxSDP, 421
2501
提取空间
417
305
CPRC
319
QME
401
217
转发描述符
接收信道处理器 307(i)
描述符 217
队列
215
发送信道处理器 307(i)
305
CPRC
317
QME
401
转发描述符
融合空间
419
405
229
向外发送串行数据流
420
2505
231
317
数据缓冲器
2503
206(j)
串行数据处理器
DMA
DMEM
DMA
BME
包缓冲器
TxSDP, 427
315
外部SDRAM

图 25

图 26

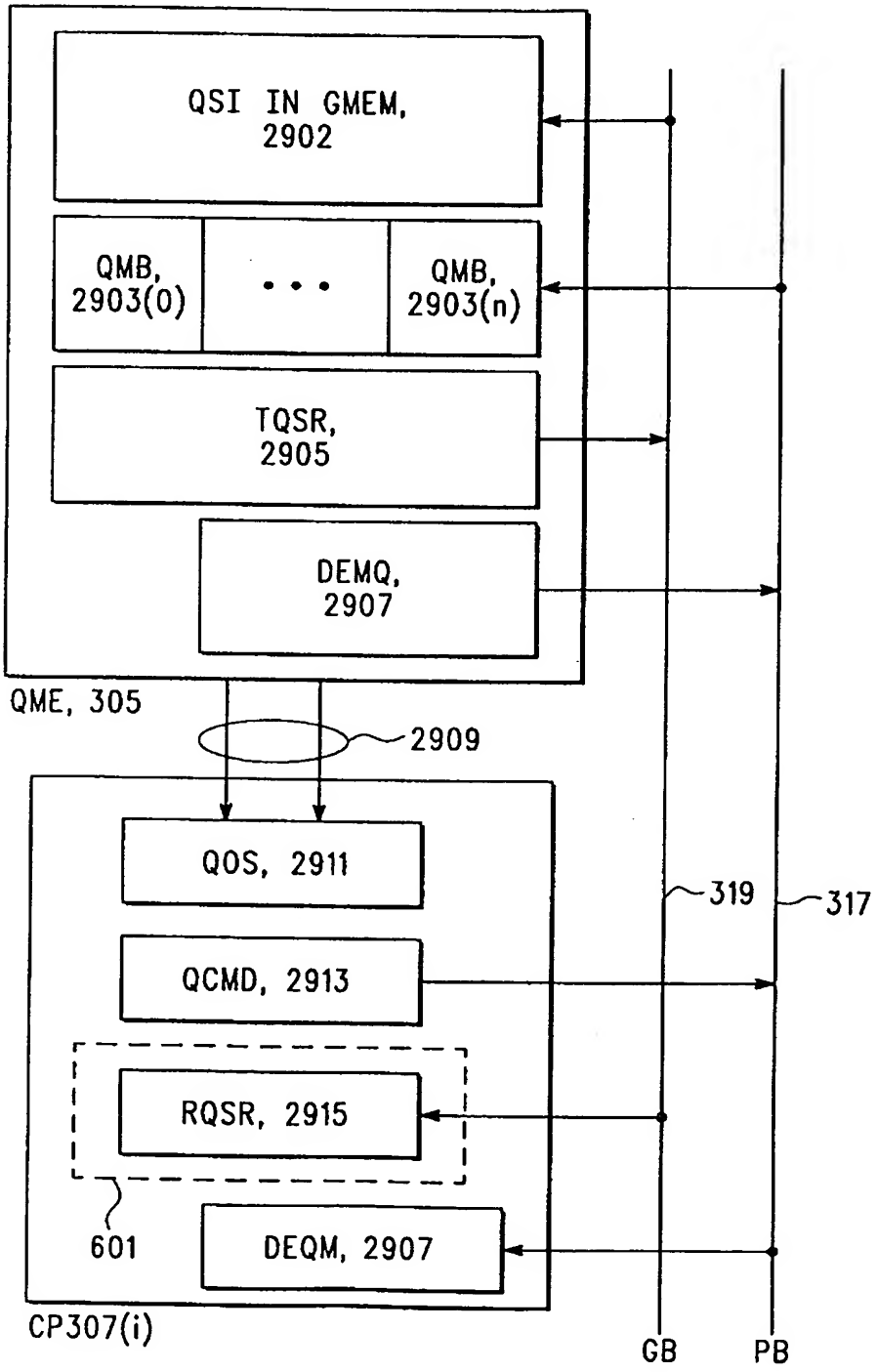

1027
RxBIT
令牌总线
环型总线接口
RxSDP 3
串行时钟
核心
时钟
2601
1021
1023
比特提取
1003
1004
1011
1028
1001
1005
1015
1013
小 FIFO
大 FIFO
8B
4B
8B
64B
128B
128B
提取/融合空回
415
409
417
425
10
8b/10b
解码
RxBIT
处理器
16 字节
缓冲器
1025
3
同步丢失
421(i,3)
2603
421(i,2)
RxSDP 2
421(i,1)
RxSDP 1
421(i,0)
RxSDP 0
PHY
可配置令牌逻辑
Rx 字节令牌总线


TxBYTE令牌总线
1225
TxSDP
427(j,3)
TxSDP
427(j,2)
TxSDP
427(j,1)
TxSDP
427(j,0)
PHY
可配置管脚逻辑
大 FIFO
TxBIT 处理器
16 字节缓冲器
8B
8B
64B
128B
128B
提取/融合空间
来自 SDP
的 1，2，和 3 的组合路径
串行时钟
核心时钟
1208
1223
1211
1021
1023
1203
1201
1205
1213
1215
1229
419
431
10
206(i)
8b/10b 解码
小 FIFO
RxBIT 处理器
MUX
TxBYTE TOKEN BUS

2701 图 27


2805
2807
2809
2811
2813
2815
CTL,
2803
M
TY
LEN
SEQ
DEST
SRC
数据，2817
环型总线消息，2801

图 28


QSI IN GMEM, 2902
QMB, 2903(0)
QMB, 2903(n)
TQSR, 2905
DEMQ, 2907
QME, 305
2909
QOS, 2911
QCMD, 2913
RQSR, 2915
601
DEQM, 2907
CP307(i)
319
317
GB
PB

2901

图 29

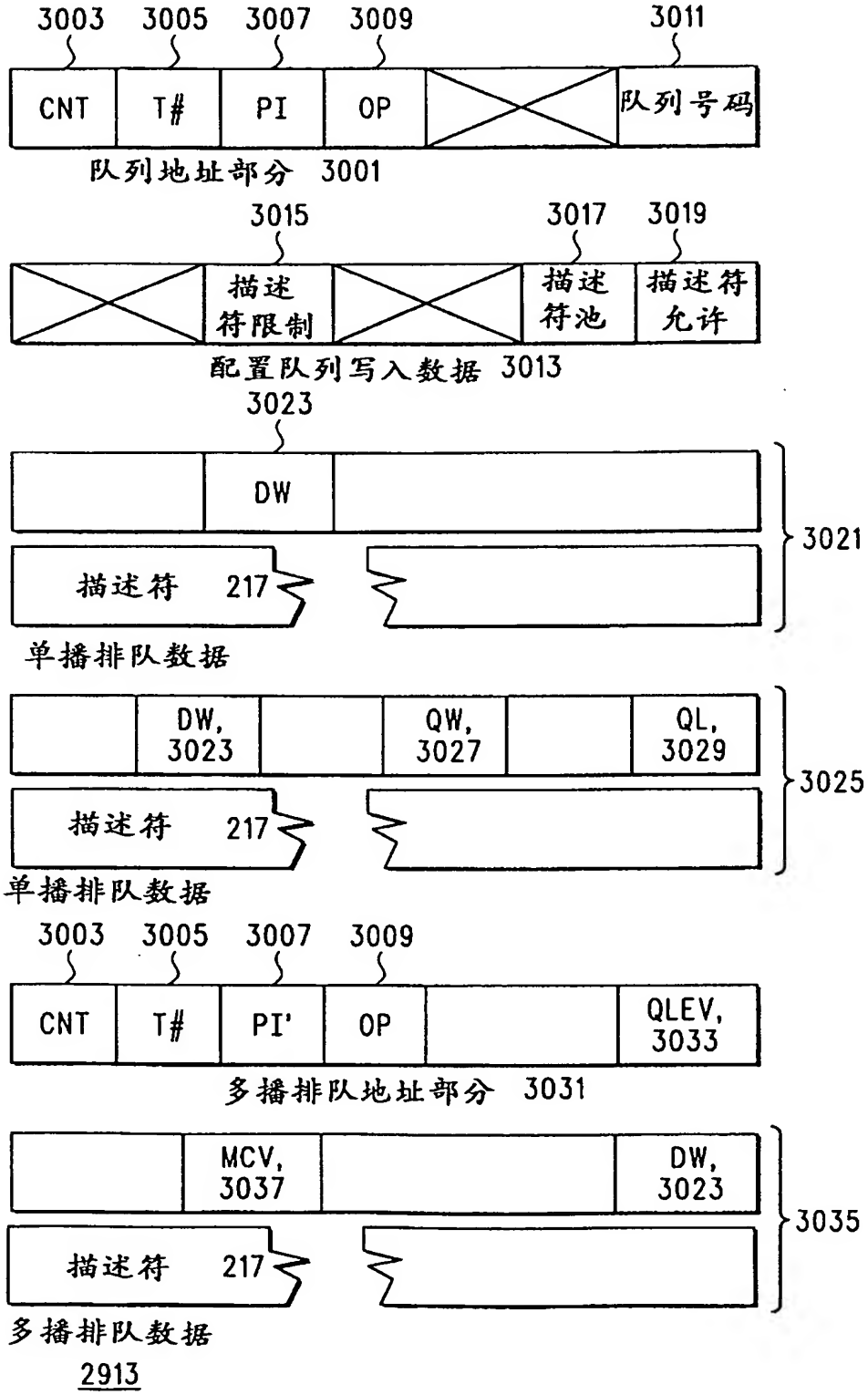

3003
3005
3007
3009
3011
CNT
T#
PI
OP
队列号码
队列地址部分 3001
3015
3017
3019
描述符限制
描述符池
描述符允许
配置队列写入数据 3013
3023
DW
3021
描述符 217
单播排队数据
DW, 3023
QW, 3027
QL, 3029
3025
描述符 217
单播排队数据
3003
3005
3007
3009
CNT
T#
PI'
OP
QLEV, 3033
多播排队地址部分 3031
MCV, 3037
DW, 3023
3035
描述符 217
多播排队数据
2913

图 30

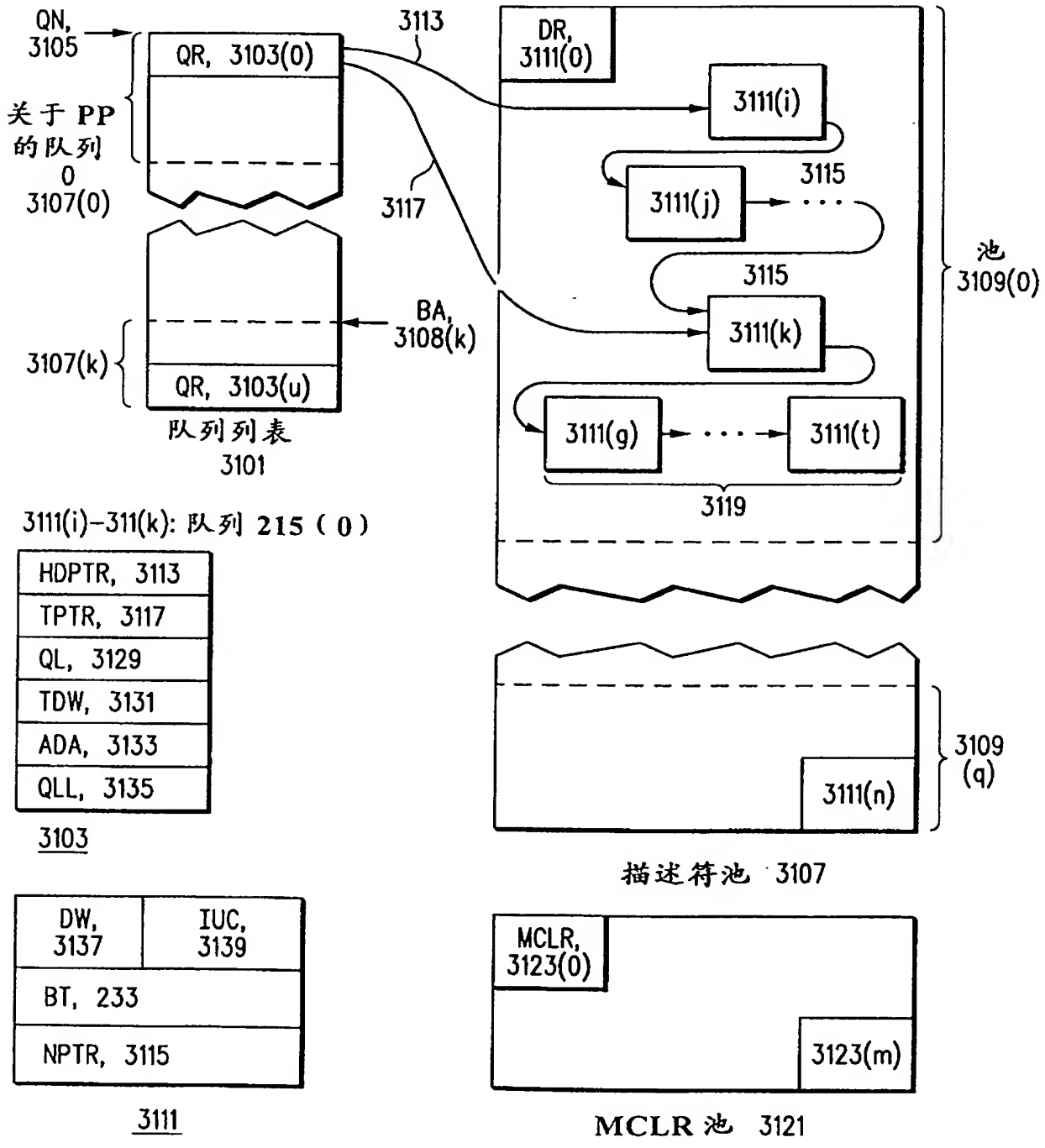

QN, 3105
关于 PP 的队列 0 3107(0)
QR, 3103(0)
3113
3117
DR, 3111(0)
3111(i)
3111(j)
3115
3115
3111(k)
池 3109(0)
BA, 3108(k)
3107(k)
QR, 3103(u)
队列列表 3101
3111(g)
3111(t)
3119
3111(i)-311(k): 队列 215 ( 0 )
HDPTR, 3113
TPTR, 3117
QL, 3129
TDW, 3131
ADA, 3133
QLL, 3135
3103
3111(n)
3109 (q)
描述符池 3107
DW, 3137
IUC, 3139
BT, 233
NPTR, 3115
3111
MCLR, 3123(0)
3123(m)
MCLR 池 3121

图 31

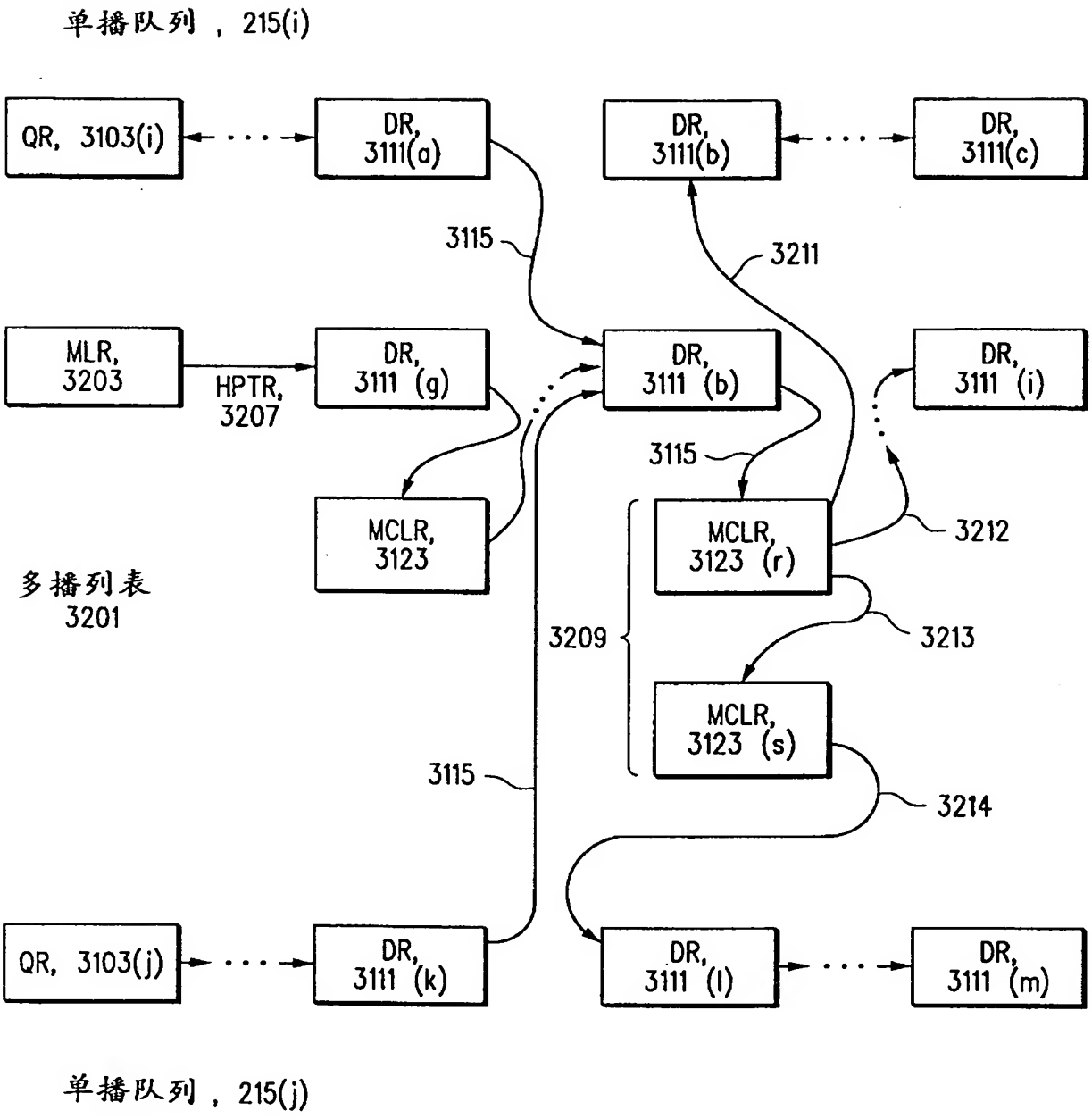

单播队列，215(i)
QR, 3103(i)
DR, 3111(a)
DR, 3111(b)
DR, 3111(c)
3115
3211
MLR, 3203
HPTR, 3207
DR, 3111 (g)
DR, 3111 (b)
DR, 3111 (i)
3115
MCLR, 3123
MCLR, 3123 (r)
3212
多播列表 3201
3209
3213
MCLR, 3123 (s)
3115
3214
QR, 3103(j)
DR, 3111 (k)
DR, 3111 (l)
DR, 3111 (m)
单播队列，215(j)

图 32

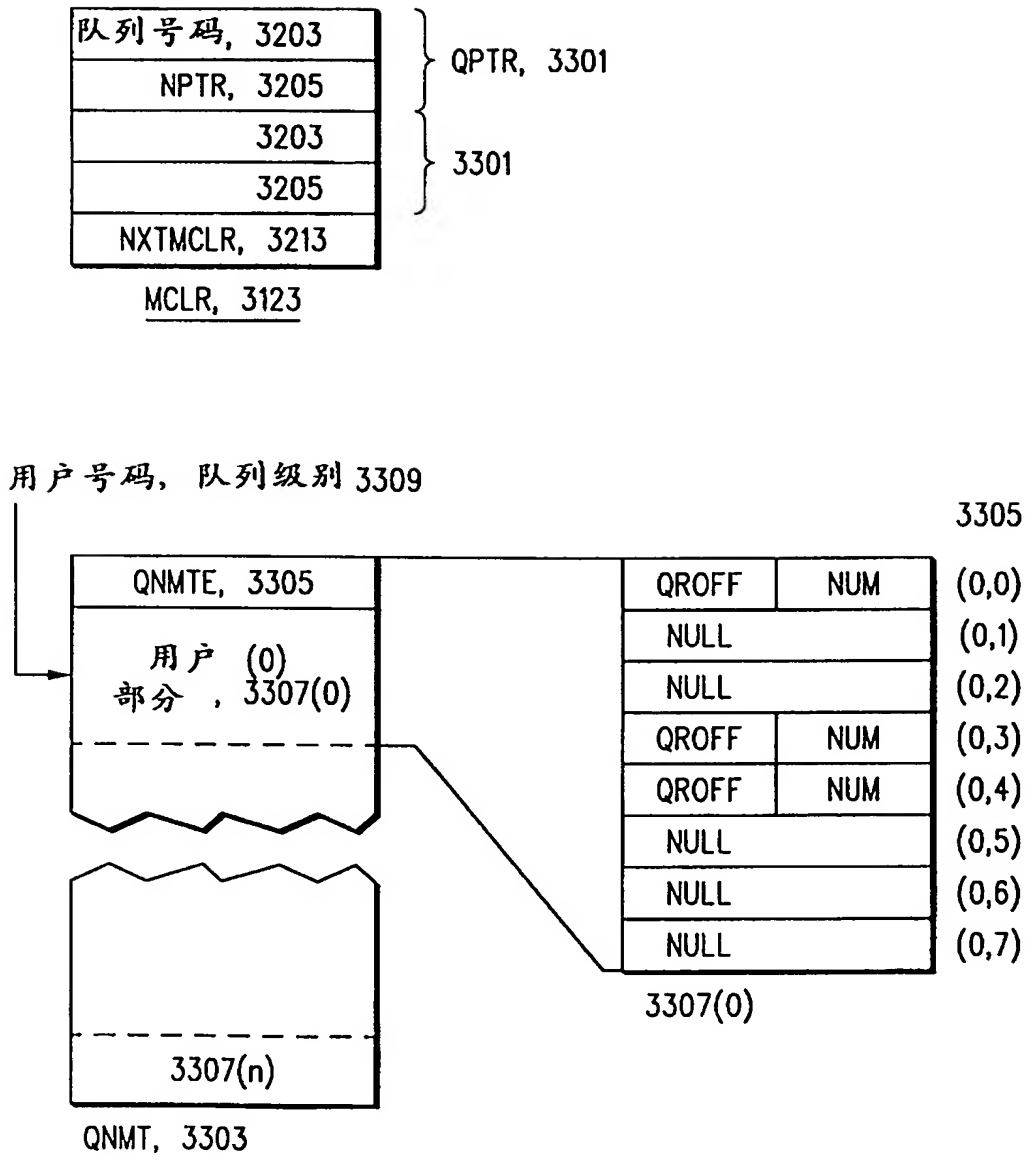

队列号码, 3203
NPTR, 3205
3203
3205
NXTMCLR, 3213
MCLR, 3123
QPTR, 3301
3301
用户号码, 队列级别 3309
3305
QNMTE, 3305
用户 (0)
部分 , 3307(0)
3307(n)
QNMT, 3303
QROFF
NUM
NULL
(0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
3307(0)

图 33

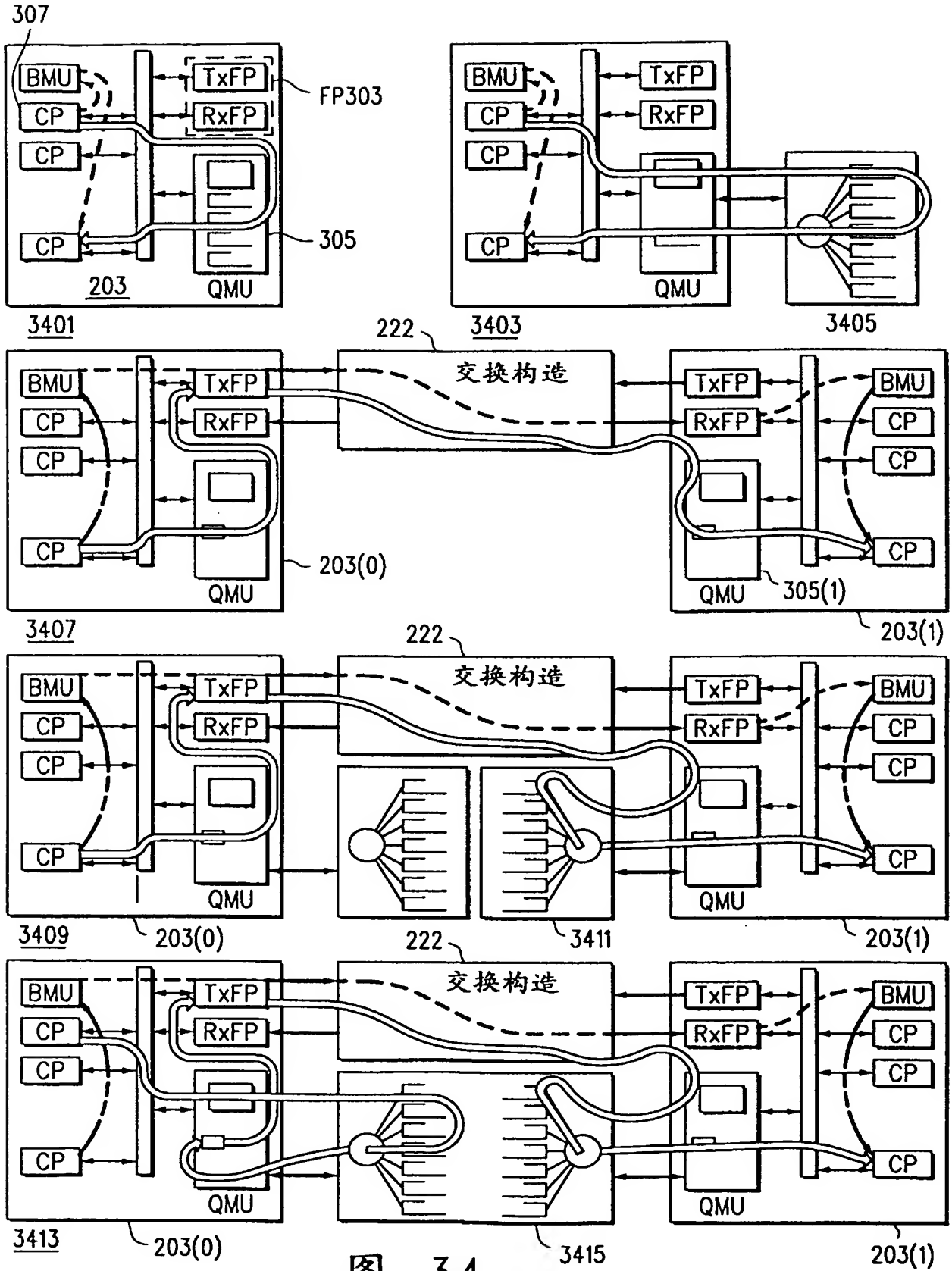
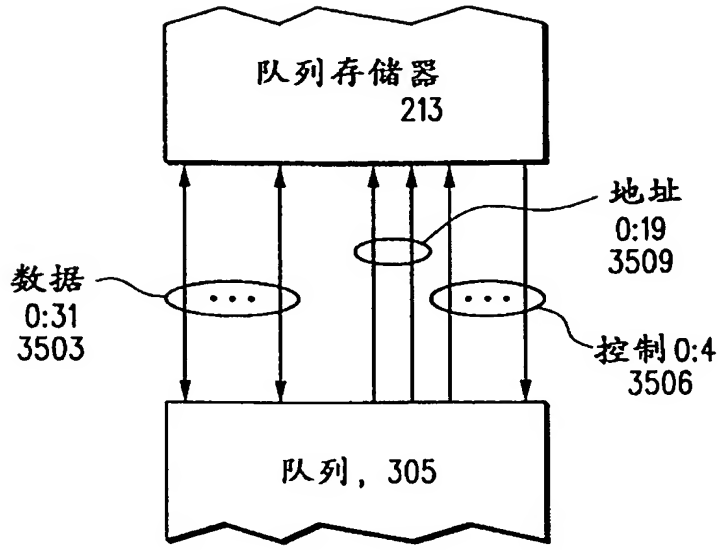
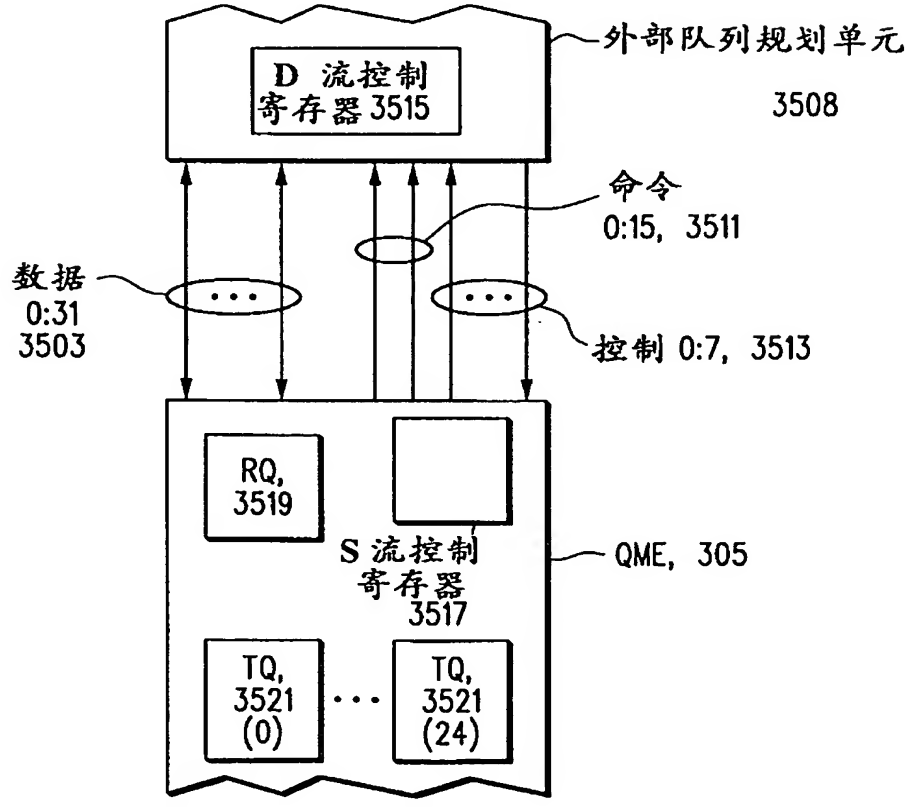

307
BMU
CP
CP
CP
TxFP
RxFP
FP303
305
203
QMU
3401
3403
3405
222
交换构造
3407
203(0)
305(1)
203(1)
3409
3411
3413
3415

图 34

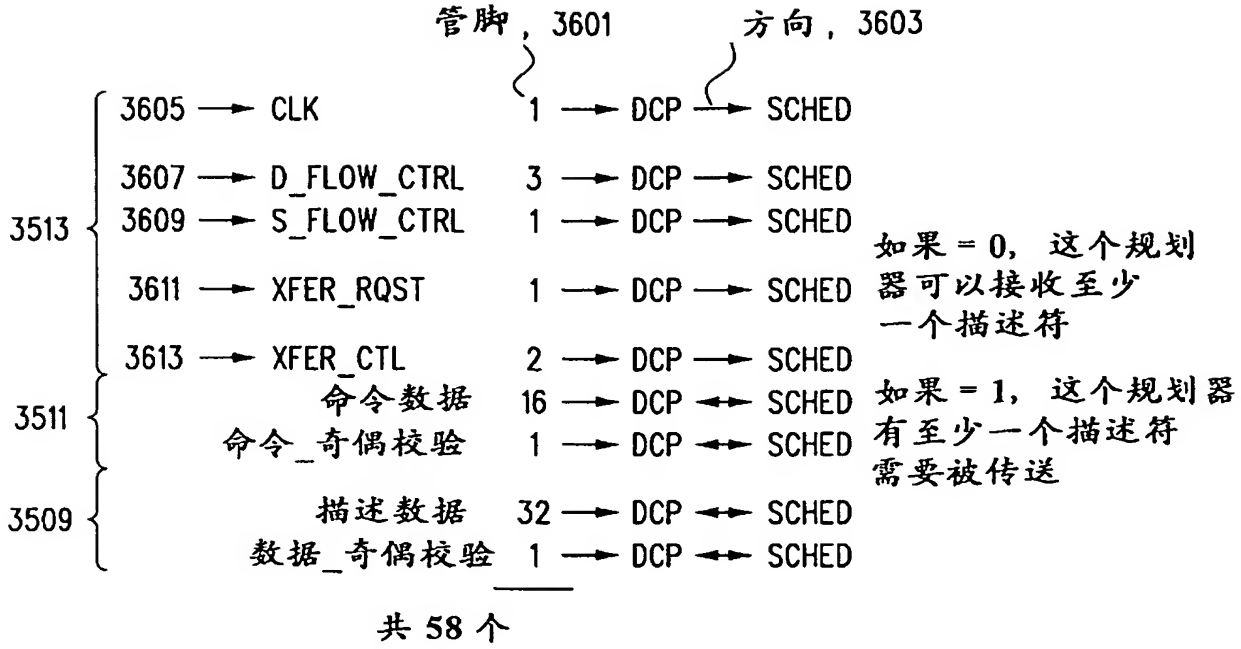



存储器外部接口，3501




规划器外部接口，3507

图 35


管脚，3601
方向，3603
3605 CLK 1 DCP SCHED
3607 D_FLOW_CTRL 3 DCP SCHED
3513
3609 S_FLOW_CTRL 1 DCP SCHED
如果 = 0，这个规划器可以接收至少一个描述符
3611 XFER_RQST 1 DCP SCHED
3613 XFER_CTL 2 DCP SCHED
3511
命令数据 16 DCP SCHED
命令_奇偶校验 1 DCP SCHED
如果 = 1，这个规划器有至少一个描述符需要被传送
3509
描述数据 32 DCP SCHED
数据_奇偶校验 1 DCP SCHED
共 58 个
3507

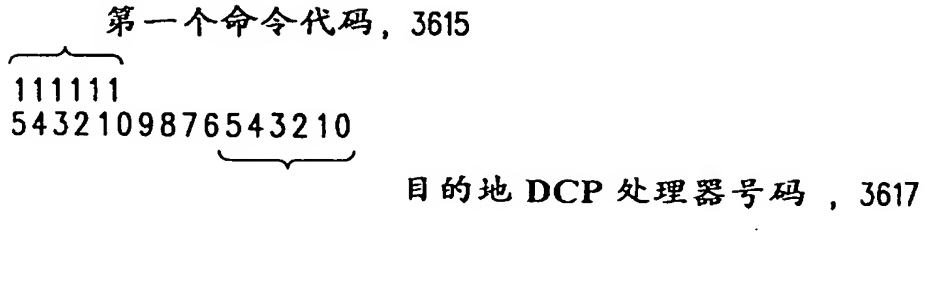

第一个命令代码，3615
111111
5432109876543210
目的地 DCP 处理器号码，3617
3514

图 36

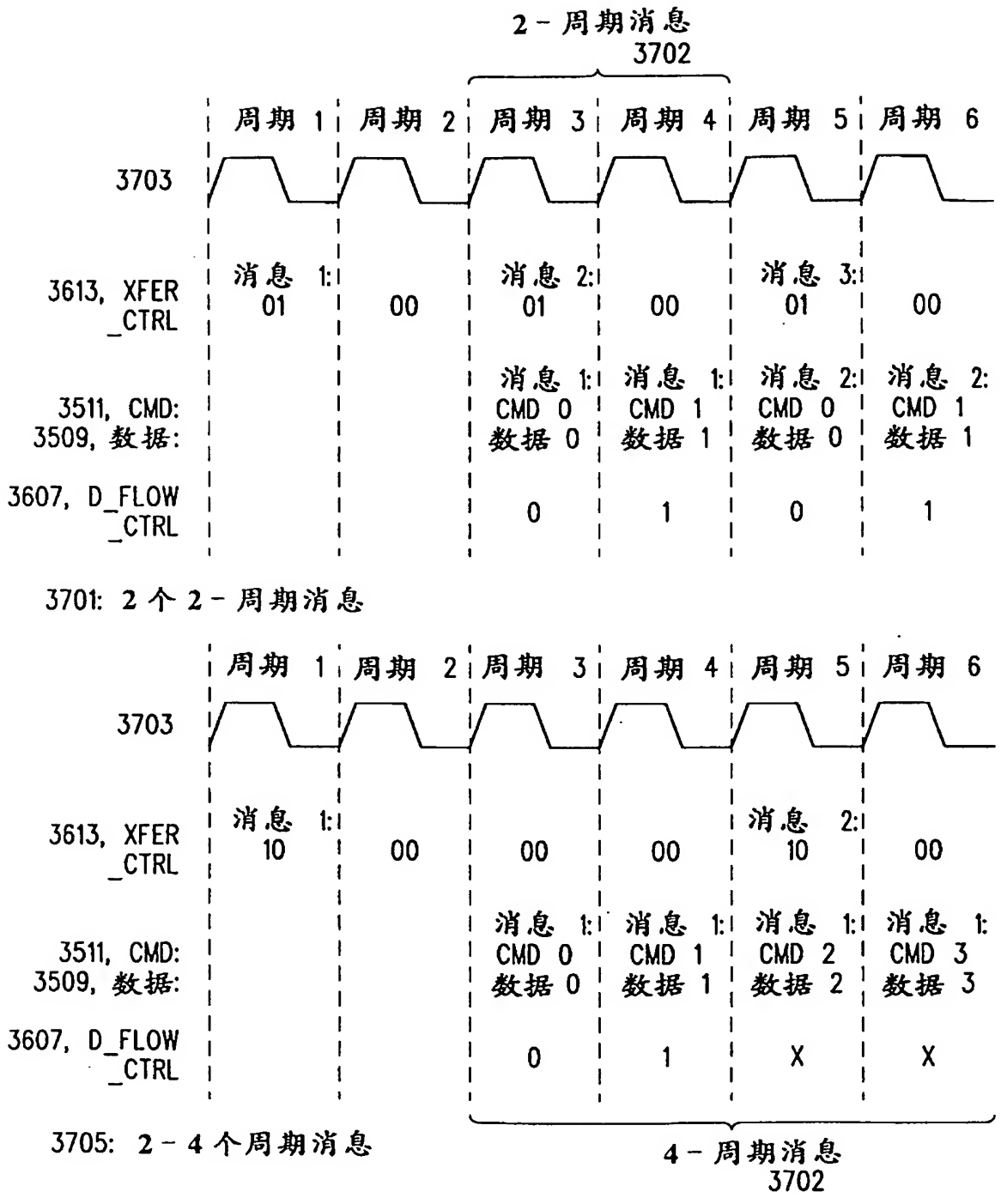

2 - 周期消息
3702
周期 1
周期 2
周期 3
周期 4
周期 5
周期 6
3703
3613, XFER _CTRL
消息 1: 01
00
消息 2: 01
00
消息 3: 01
00
3511, CMD:
3509, 数据:
消息 1: CMD 0 数据 0
消息 1: CMD 1 数据 1
消息 2: CMD 0 数据 0
消息 2: CMD 1 数据 1
3607, D_FLOW _CTRL
0
1
0
1
3701: 2 个 2 - 周期消息
周期 1
周期 2
周期 3
周期 4
周期 5
周期 6
3703
3613, XFER _CTRL
消息 1: 10
00
00
00
消息 2: 10
00
3511, CMD:
3509, 数据:
消息 1: CMD 0 数据 0
消息 1: CMD 1 数据 1
消息 1: CMD 2 数据 2
消息 1: CMD 3 数据 3
3607, D_FLOW _CTRL
0
1
X
X
3705: 2 - 4 个周期消息
4 - 周期消息
3702

图 37

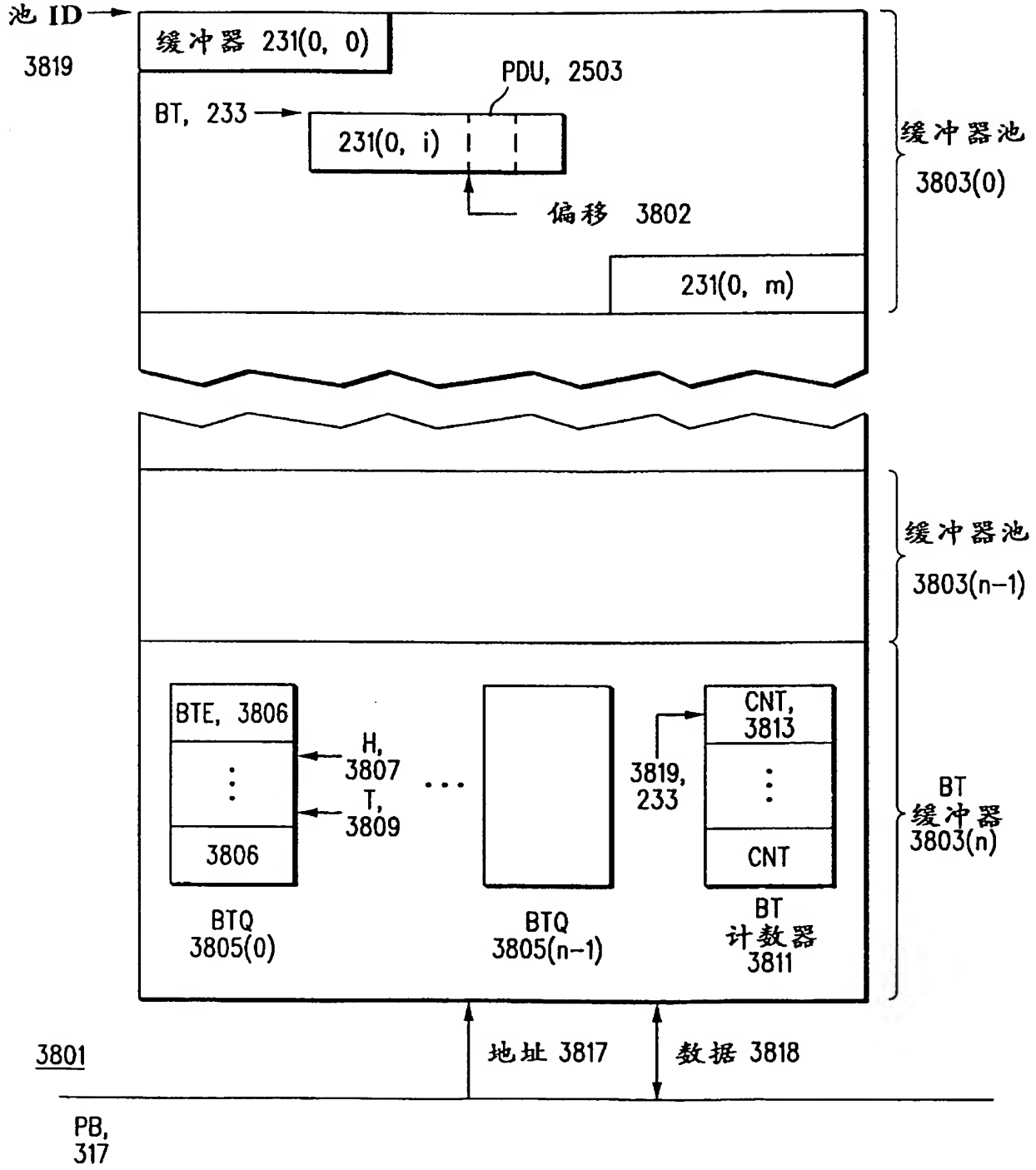
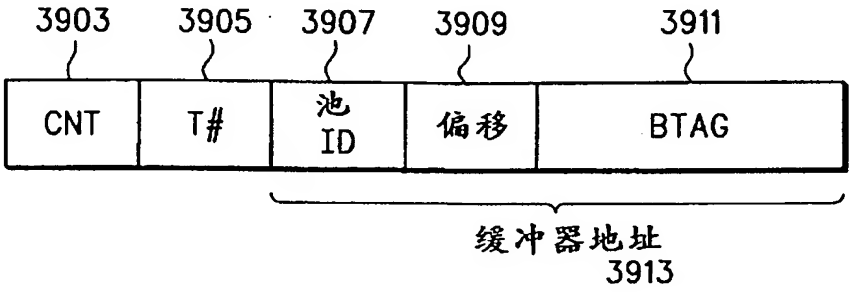
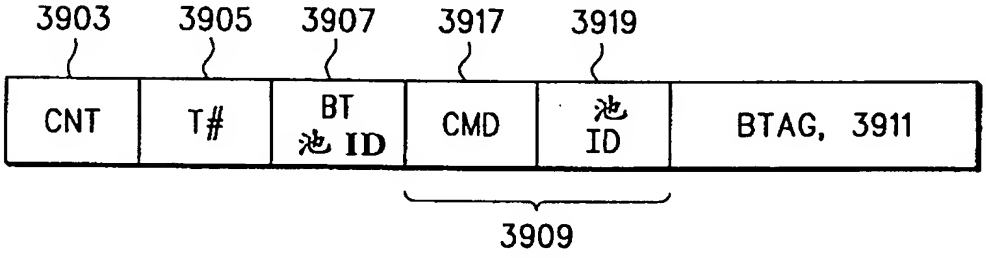

池 ID
3819
缓冲器 231(0, 0)
PDU, 2503
BT, 233
231(0, i)
偏移 3802
231(0, m)
缓冲器池
3803(0)
缓冲器池
3803(n-1)
BTE, 3806
H, 3807
T, 3809
3806
BTQ
3805(0)
BTQ
3805(n-1)
CNT, 3813
3819, 233
CNT
BT
计数器
3811
BT
缓冲器
3803(n)
3801
地址 3817
数据 3818
PB, 317

图 38




负荷总线缓冲器读取/写入命令 3901




负荷总线 BTAG 命令, 3915

图 39

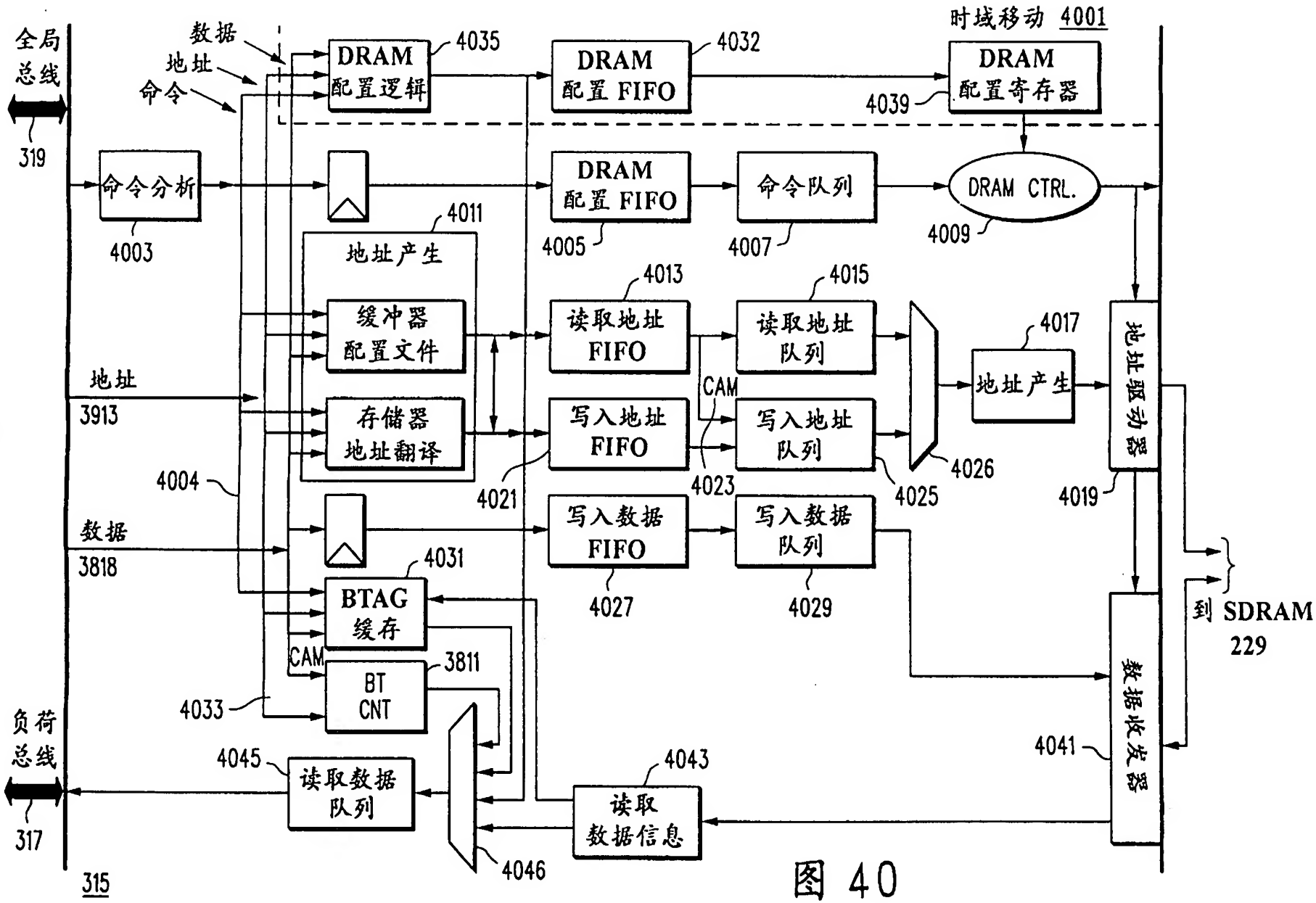
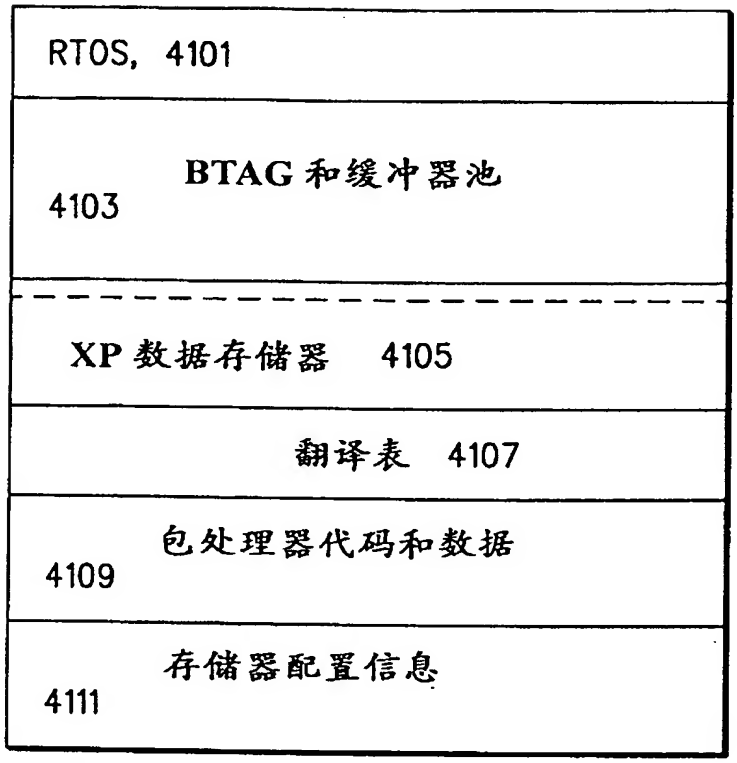
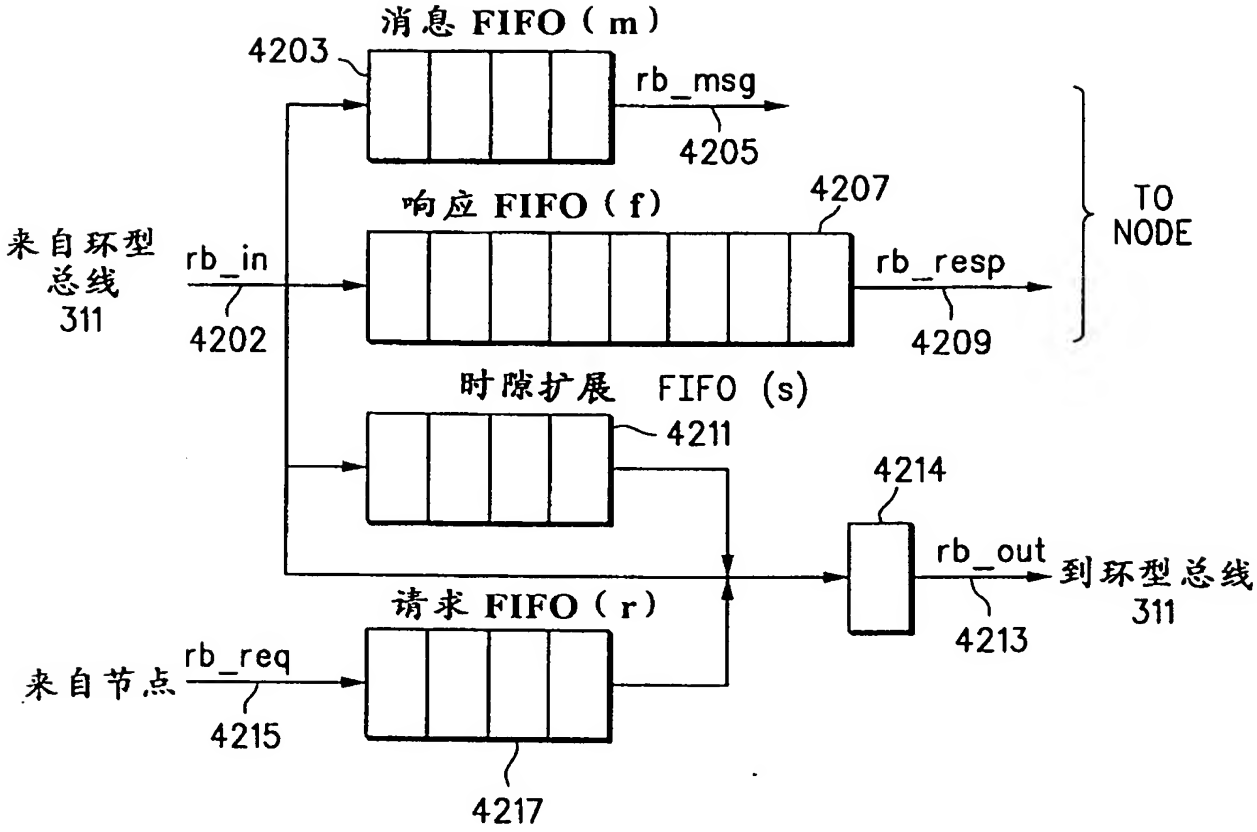

全局
总线
319
数据
地址
命令
DRAM
配置逻辑
4035
DRAM
配置 FIFO
4032
时域移动 4001
DRAM
配置寄存器
4039
命令分析
4003
DRAM
配置 FIFO
4005
命令队列
4007
DRAM CTRL.
4009
4011
地址产生
缓冲器
配置文件
存储器
地址翻译
4013
读取地址
FIFO
4015
读取地址
队列
4017
地址产生
地址驱动器
4019
地址
3913
CAM
写入地址
FIFO
写入地址
队列
4026
4025
4023
4021
4004
数据
3818
写入数据
FIFO
写入数据
队列
4027
4029
4031
BTAG
缓存
CAM
BT
CNT
3811
4033
负荷
总线
317
4045
读取数据
队列
4043
读取
数据信息
4046
数据收发器
4041
到 SDRAM
229
315

图 40


RTOS, 4101
BTAG 和缓冲器池
4103
XP 数据存储器 4105
翻译表 4107
包处理器代码和数据
4109
存储器配置信息
4111

229

图 41




环型总线节点接口
4201

图 42

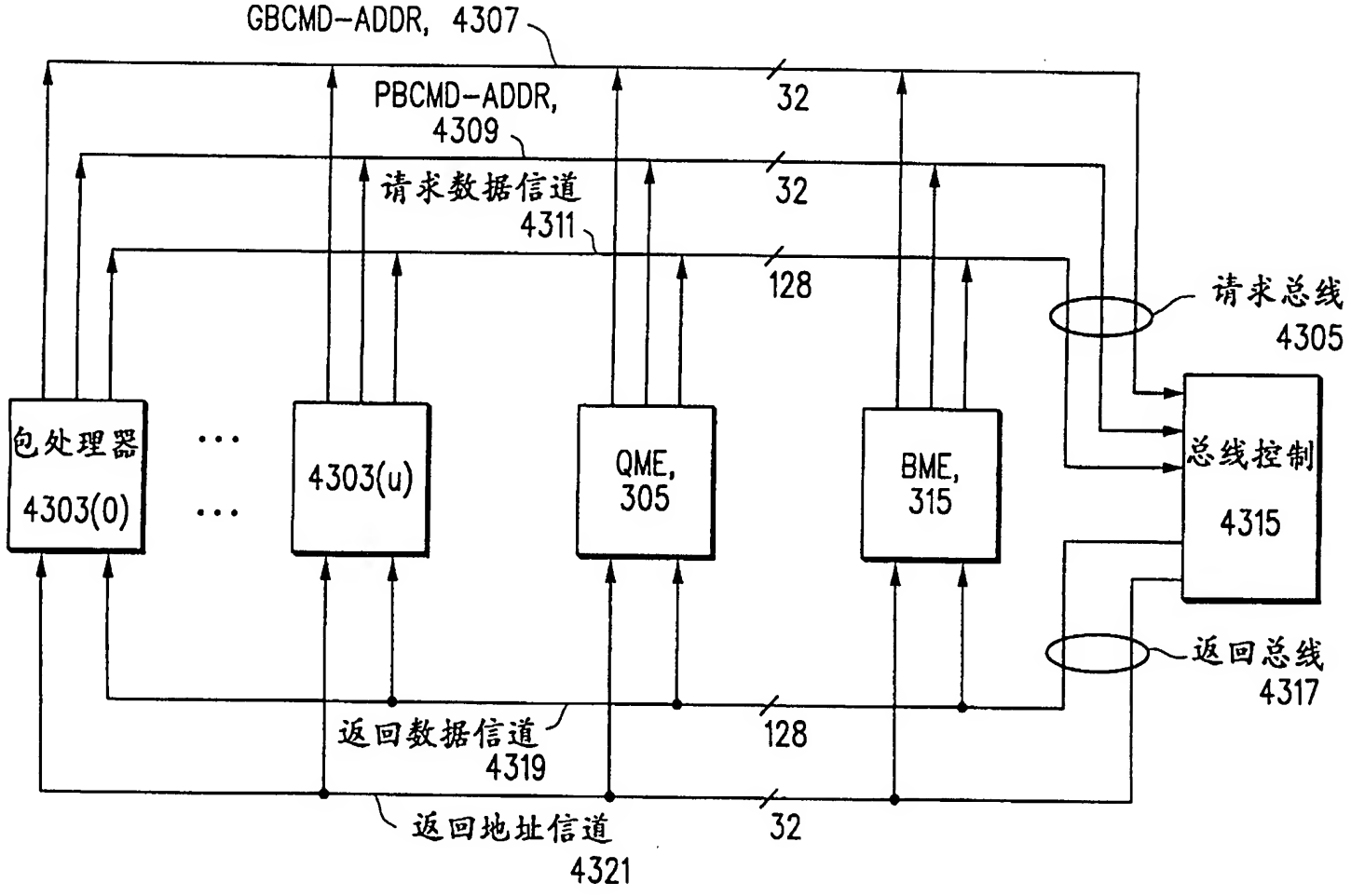

GBCMD-ADDR, 4307
PBCMD-ADDR, 4309
32
请求数据信道 4311
32
128
请求总线 4305
包处理器 4303(0)
4303(u)
QME, 305
BME, 315
总线控制 4315
返回总线 4317
返回数据信道 4319
128
返回地址信道 4321
32
4301

图 43

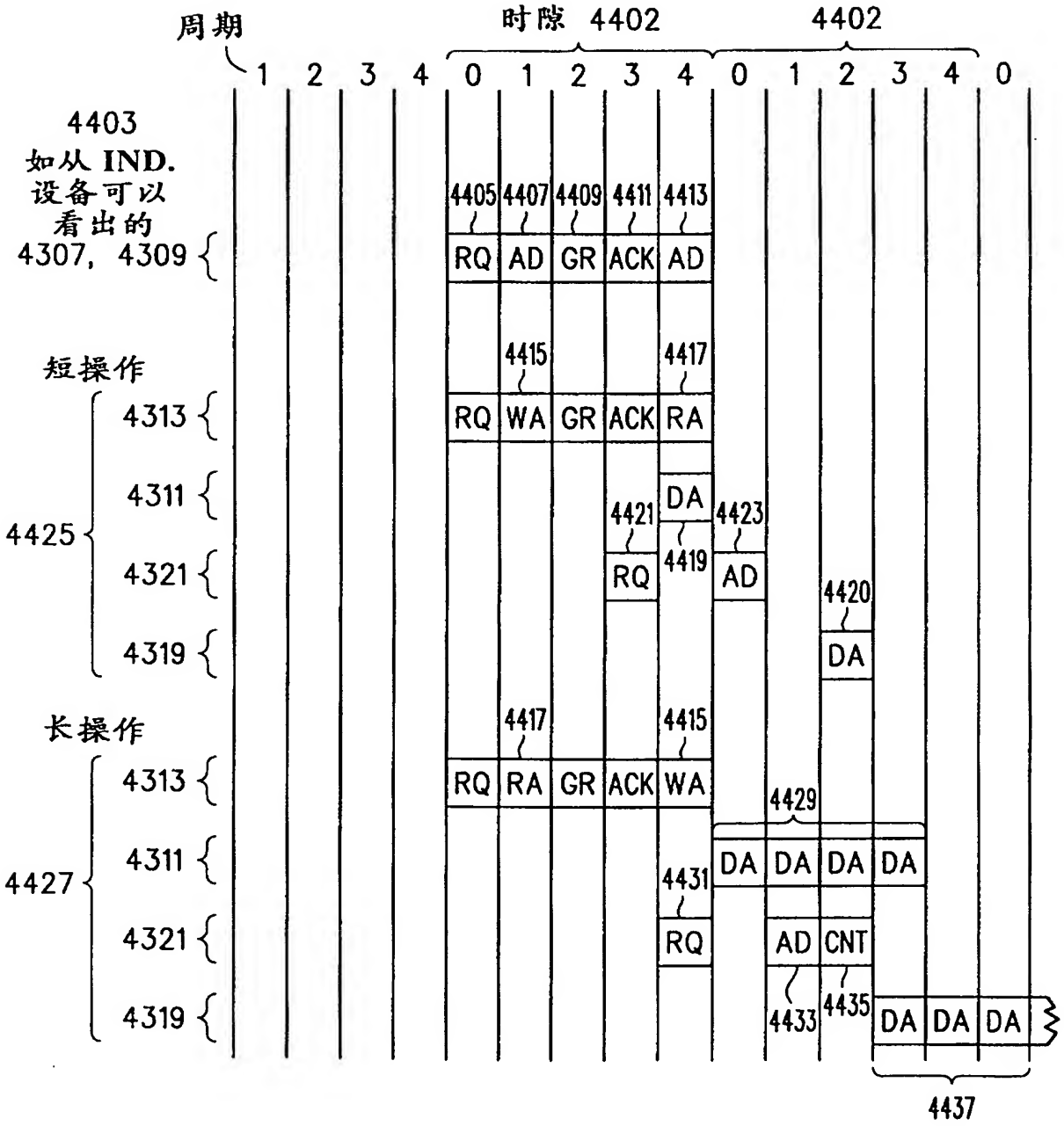
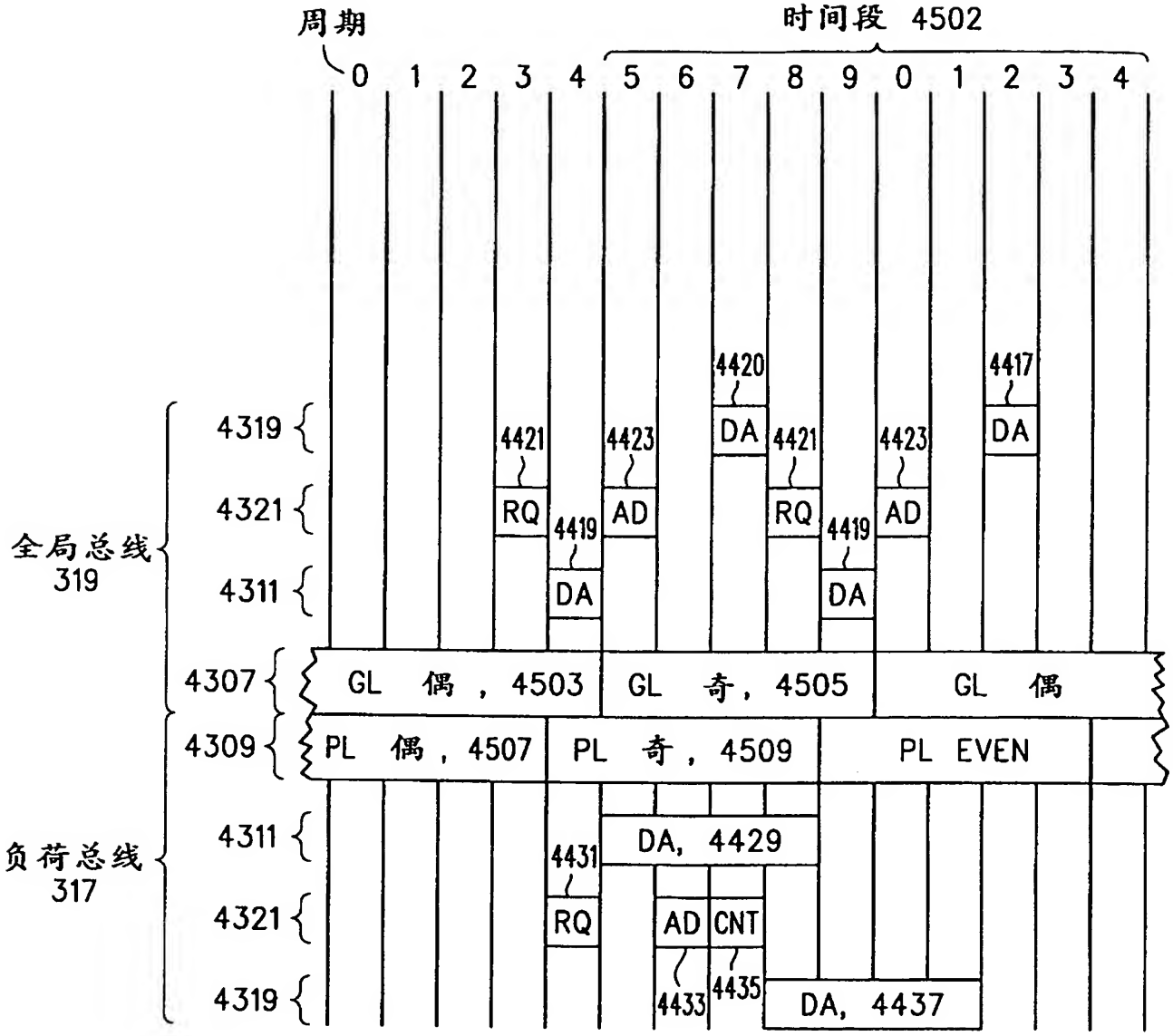

周期
1 2 3 4
时隙 4402
0 1 2 3 4
4402
0 1 2 3 4 0
4403
如从 IND.
设备可以
看出的
4307, 4309
4405 4407 4409 4411 4413
RQ AD GR ACK AD
短操作
4415 4417
4313
RQ WA GR ACK RA
4311
4421 DA 4423
4425
4321
RQ 4419 AD
4420
4319
DA
长操作
4417 4415
4313
RQ RA GR ACK WA
4429
4311
DA DA DA DA
4427
4431
4321
RQ AD CNT
4319
4433 4435 DA DA DA
4437
4401

图 44


周期
时间段 4502
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
全局总线 319
负荷总线 317
4319
4321
4311
4307
4309
4311
4321
4319
4421
4423
4420
4417
4419
4431
4433
4435
RQ
AD
DA
CNT
GL 偶，4503
GL 奇，4505
GL 偶
PL 偶，4507
PL 奇，4509
PL EVEN
DA，4429
DA，4437

4501

图 45

| 4603 | 4605 | 4607 | 4609 | 4611 | 4613 | 4615 | | 4617 | | 4619 |
|---|---|---|---|---|---|---|---|---|---|---|
| 管脚 | 目的 | RMII | OC-3 | DS1 | DS3 | GMII (Tx) | GMII (Rx) | TBI (Tx) | TBI (Rx) | OC-12 |
| CP0_0 | OUTCLK | REF_CLK | RCLK_H | TCLK | TCLK | TCLK | nc | TCLK | nc | TCLK |
| _1 | INCLK | CRS_DV | RCLK_L | RCLK | RCLK | CRS | nc | | nc | TCLK1 |
| _2 | 数据 | TXD(0) | TXD_H | TDATA | TDATA | TXD(0) | nc | TXD(0) | nc | TDX(0) |
| _3 | 数据 | TXD(1) | TXD_L | TxFRAME | TxFRAME | TXD(1) | nc | TXD(1) | nc | TDX(1) |
| _4 | 数据 | RXD(0) | RXD_H | RDATA | RDATA | TXD(2) | nc | TXD(2) | nc | TDX(2) |
| _5 | 数据 | RXD(1) | RXD_L | RxFRAME | RxFRAME | TXD(3) | nc | TXD(3) | nc | TDX(3) |
| _6 | 数据 | TX-EN | SIGNAL_DET | | | TX_EN | nc | TXD(4) | nc | |
| CP1_0 | OUTCLK | REF_CLK | RCLK_H | TCLK | TCLK | | | | | |
| _1 | INCLK | CRS_DV | RCLK_L | RCLK | RCLK | COL | nc | | | |
| _2 | 数据 | TXD(0) | TXD_H | TDATA | TDATA | TXD(4) | nc | TXD(5) | nc | TDX(4) |
| _3 | 数据 | TXD(1) | TXD_L | TxFRAME | TxFRAME | TXD(5) | nc | TXD(6) | nc | TDX(5) |
| _4 | 数据 | RXD(0) | RXD_H | RDATA | RDATA | TXD(6) | nc | TXD(7) | nc | TDX(6) |
| _5 | 数据 | RXD(1) | RXD_L | RxFRAME | RxFRAME | TXD(7) | nc | TXD(8) | nc | TDX(7) |
| _6 | 数据 | TX-EN | SIGNAL_DET | | | TX_ER | nc | TXD(9) | nc | |
| CP2_0 | OUTCLK | REF_CLK | RCLK_H | TCLK | TCLK | | | | | |
| _1 | INCLK | CRS_DV | RCLK_L | RCLK | RCLK | nc | RCLX | nc | RCLK | RCLK1 |
| _2 | 数据 | TXD(0) | TXD_H | TDATA | TDATA | nc | RXD(0) | nc | RXD(1) | RDX(0) |
| _3 | 数据 | TXD(1) | TXD_L | TxFRAME | TxFRAME | nc | RXD(1) | nc | RXD(0) | RDX(1) |
| _4 | 数据 | RXD(0) | RXD_H | RDATA | RDATA | nc | RXD(2) | nc | RXD(2) | RDX(2) |
| _5 | 数据 | RXD(1) | RXD_L | RxFRAME | RxFRAME | nc | RXD(3) | nc | RXD(3) | RDX(3) |
| _6 | 数据 | TX-EN | SIGNAL_DET | | | nc | RX_DV | | RXD(8) | FP |
| CP3_0 | OUTCLK | REF_CLK | RCLK_H | TCLK | TCLK | | | | | |
| _1 | INCLK | CRS_DV | RCLK_L | RCLK | RCLK | | | nc | RCLKN | |
| _2 | 数据 | TXD(0) | TXD_H | TDATA | TDATA | nc | RXD(4) | nc | RXD(4) | RDX(4) |
| _3 | 数据 | TXD(1) | TXD_L | TxFRAME | TxFRAME | nc | RXD(5) | nc | RXD(5) | RDX(5) |
| _4 | 数据 | RXD(0) | RXD_H | RDATA | RDATA | nc | RXD(6) | nc | RXD(6) | RDX(6) |
| _5 | 数据 | RXD(1) | RXD_L | RxFRAME | RxFRAME | nc | RXD(7) | nc | RXD(7) | RDX(7) |
| _6 | 数据 | TX-EN | SIGNAL_DET | | | nc | RX_ER | nc | RXD(9) | LOCKDET |

4601

图 46

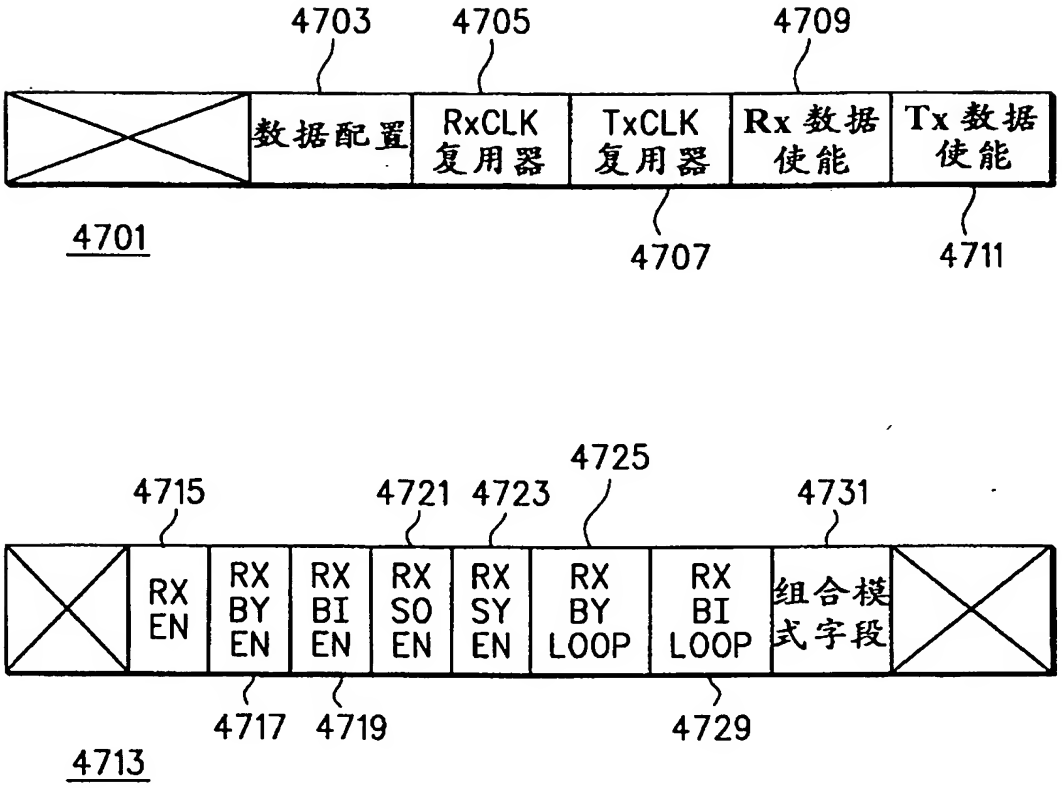

4703
4705
4709
数据配置
RxCLK 复用器
TxCLK 复用器
Rx 数据使能
Tx 数据使能
4701
4707
4711
4725
4715
4721
4723
4731
RX EN
RX BY EN
RX BI EN
RX SO EN
RX SY EN
RX BY LOOP
RX BI LOOP
组合模式字段
4717
4719
4729
4713

图 47